Western Graduate&PostdoctoralStudies

Western University
Scholarship@Western

Electronic Thesis and Dissertation Repository

4-28-2020 11:30 AM

Range Flow: New Algorithm Design and Quantitative and Qualitative Analysis

Seereen Noorwali
The University of Western Ontario

Graduate Program in Computer Science
A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy
© Seereen Noorwali 2020

Follow this and additional works at: https://ir.lib.uwo.ca/etd

 Part of the Computer Sciences Commons

Recommended Citation

Noorwali, Seereen, "Range Flow: New Algorithm Design and Quantitative and Qualitative Analysis" (2020). *Electronic Thesis and Dissertation Repository*. 6991.
https://ir.lib.uwo.ca/etd/6991

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.

Abstract

Optical flow computation is one of the oldest and most active research fields in computer vision and image processing. It encompasses the following areas: motion estimation, video compression, object detection and tracking, image dominant plane extraction, movement detection, robot navigation, visual odometry, traffic analysis, and vehicle tracking. Optical flow methods calculate the motion between two image frames. In 2D images, optical flow specifies how far each pixel moves between adjacent frames; in 3D images, it specifies how much each voxel moves between adjacent volumes in the dataset. Since 1980, several algorithms have successfully estimated 2D and 3D optical flow. Notably, scene flow and range flow are special cases of 3D optical flow. Scene flow is the 3D optical flow of pixels on a moving surface. Scene flow uses disparity and disparity gradient maps computed from a stereo sequence and the 2D optical flow of the left and right images in the stereo sequence to compute 3D motion. Range flow is similar to scene flow, but is calculated from depth map sequences or range datasets. There is clear overlap between the algorithms that compute scene flow and range flow. Therefore, we propose new insights that can help range flow algorithms to advance to the next stage. We propose new insights into range flow algorithms by enhancing them to allow large displacements using a hierarchical framework with warping technique. We applied robust statistical formulations to generate robust and dense flow to overcome motion discontinuities and reduce the outliers. Overall, this thesis focuses on the estimation of 2D optical flow and 3D range flow using several algorithms. In addition, we studied depth data gained from different sensors and cameras. These cameras provided RGB-D data that allowed us to compute 3D range flow in two ways: using depth data only, or by combining intensity with depth data to improve the flow. We implemented well-known local approaches LK [1] and global HS [2] algorithms and recast them in the proposed framework to estimate 2D and 3D range flow [3]. Furthermore, combining local and global algorithm (CLG) proposed by Bruhn et al. [4, 5] as well as Brox et al. [6] method are implemented to estimate 2D optical flow and 3D range flow. We tested and evaluated these implemented approaches both qualitatively and quantitatively in two different motions (translation and divergence) using several real datasets acquired using Kinect V2, ZED camera, and iPhone X (front and rear) Cameras. We found that CLG and Brox methods gave the best results in our datasets using Kinect V2, ZED and front camera in iPhone X sequences.

Keywords: Optical flow, 3D range flow, Scene flow, Depth information, Range imaging, Depth sensor/camera, Kinect V2, ZED , iPhone X, Hierarchical framework, Quantitative and qualitative error analysis

Summary For Lay Audience

Optical flow can be defined as the estimation of motion in a set of image sequences. It can be computed from 2D data, which are regular images captured by typical cameras, to estimate 2D optical flow. 2D optical flow gives (u,v) motion in the image, which describes how much the object moves in the x-direction (u) and the y-direction (v). However, using 3D data, we can now estimate 3D optical flow (u,v,w). The first two components (u,v) describe the objects motion in x and y-direction, respectively. The third component (w) describes the objects motion in the z-direction. This study shows how to estimate a special kind of 3D flow, which is the motion of a moving objects surface. This type of flow is a Scene, or Range, flow. This particular flow needs 2D data in addition to depth information from the scene, or image. The depth information calculates the distance of the object from the camera as measured in units such as millimetres. This depth can be extracted using special kinds of cameras and sensors. In this thesis, we estimated 2D optical flow and 3D range flow using six differing classic approaches. These six approaches include: Least Squared (**LS**), Total Least Squared (**TLS**), Global Regularization (**Global**), Indirect Global Regularization (**Global_in**), Combined Local and Global (**CLG**), and finally the **Brox** methods. Our research extended some classic algorithms derived from 2D optical flow to produce 3D range flow. We also added some principles and parameters to better estimate both 2D and 3D flow. We tested our approaches with three different datatypes: using intensity images only, using the depth data only, and combining both depth data with intensity. We also generated new data sequences using different cameras and sensors, including Kinect V2, ZED, and iPhone X. We evaluated our approaches using various datasets and generated data, which we subsequently analyzed in both quantitative and qualitative terms.

Acknowledgements

"وَقَالُواْ ٱلْحَمْدُ لِلَّهِ ٱلَّذِى هَدَىٰنَا لِهَٰذَا وَمَا كُنَّا لِنَهْتَدِىَ لَوْلَآ أَنْ هَدَىٰنَا ٱللَّه"

All praises belong to Allah, Alhamdulillah, who has given me the health, strength, and knowledge to finish this thesis successfully. It is also my pleasure to acknowledge the Saudi government and Umm Al-Qura University for their financial support and supervision throughout my studies via the Saudi bureau in Canada.

I especially would like to thank my supervisor, Professor John Barron, for supervising me during this PhD program. Dr. Barron was both kind and fair to me throughout my five years of study. He believed in me throughout this tough journey and he supported me when I faced challenges. I really appreciate him on-going support and kindness. In particular, our weekly meetings meant a lot to me. He guided me, listened to my questions, and worked with me to solve many problems. Unfortunately, Dr. Barron passed away in 2019 before I could complete this thesis. Not only do I miss his on-going guidance and support, but I still have so many questions that I would like to ask him. He was an expert in this field and one of the only people who could truly appreciate the scope of my research here. When I had to return home to Saudi Arabia unexpectedly in the middle of my studies, Dr. Barron nevertheless pledged to support me and work despite my distance. I deeply wish that he was still here to read this thesis in its entirety and to support me on my defence date. I hope that he would be proud of the work that I accomplished under his guidance.

I also wish to extend my deepest thanks to Dr. Steven Beauchemin. He worked with me towards the end of my thesis, when I was feeling discouraged and unsure. He offered me hope, support, and time. I really appreciate his kind words about me and my work, and for supporting me through the conclusion of my thesis.

Many wonderful people in the Computer Science department at Western University also deserve a massive amount of thanks for their kindness, support, and care. In particular, I would like to thank Professors Hanan Lutfiyya, Marc Moreno Maza, and Laura Reid. I also want to extend special thanks to Janice Wiersma, the most professional secretary in the world for her patience and kind responses to my endless letter requests. I am also grateful to Professors Thomas Brox and Andres Bruhn from Germany for their comprehensive responses to my emails.

I have endless words to thank my lovely family for their prayer, support, and motivation during my PhD studies. I am especially grateful to my parents, Dr. Mohammadtaher and Sabah, my awesome grandmother, my genius brothers Dr. Abdulfattah and Dr. Hussain, and lastly, my amazing sisters, Nafessah and Safa. I am also indebted to several friends and colleagues. There are too many people to name individually, but I am grateful to each and every person who supported me before, and during, my PhD. I would like to extend special thanks to my

cousin Ibtehal, who has been my sister, friend, and colleague. I also want to thank my lovely friend Halimah who always showed such care and concern for my feelings. I hope we will stay good friends forever!

Lastly, I would like to thank myself. There were many times that I faced adversity, but I am proud that I persevered and did not quit this program or give up. I sometimes felt very lonely and discouraged because it seemed that I experienced a new problem for every year of study that I completed. During my first two years, I had some problems that almost made me quit and return to Saudi Arabia. Yet, I pushed through and stayed. My final two years of study were complicated by geopolitical tensions between Canada and Saudi Arabia, which forced me to return to Saudi Arabia for several months in 2018. When I returned in 2019 to finish this thesis, I was very sad to learn that my supervisor, Dr. Barron, had retired due to a terminal cancer diagnosis. Nevertheless, I impressed myself by finding strength I didn't know I had. I learned how to smile even when I felt sad. I tried hard to support other people even when I was struggling with my own work. I often spent so long time in my office working that I would go 24 hours or more without speaking to another person, so I had to learn how to motivate myself. Without Allah, I do not think I could have done it. Yet, I persevered, and I am grateful to myself for that. I hope I remember that if I can complete this project successfully, that I can do anything hard in my future!

Overall, this study was as challenging as it was rewarding. Generally speaking, I often found the topic of optical flow to be hard and complicated. Unfortunately, Dr. Barron's passing meant that I have not had as much feedback on this project as I would have preferred, nor had enough time to publish my work. Having to self-direct more of this project than I ever expected often made my work very difficult, because I did not always have enough support to work through challenging concepts and proper coding. Nevertheless, I am proud of what I accomplished in this thesis, and I am hopeful that it will be a significant addition to the field of optical flow.

Contents

List of Figures

List of Tables

List of Abbreviations

AAE Average Angular Error

Brox_i Brox method using Intensity data

Brox_ir Brox method using Intensity/Range data

Brox_r Brox method using Range data

CLG Combined Local Global approach

CLG_i Combined Local and Global using Intensity data

CLG_ir Combined Local and Global using Intensity/Range data

CLG_r Combined Local and Global using Range data

EPE End Point Error

Global_i Regularization using Intensity data

Global_ind_i Indirect Regularization using Intensity data

Global_ind_ir Indirect Regularization using Intensity/Range data

Global_ind_r Indirect Regularization using Range data

Global_ir Regularization using Inensity/Range data

Global_r Regularization using Range data

I Intensity image

I_t Intensity image derivative with respect to time

I_x Intensity image derivative with respect to x direction

I_y Intensity image derivative with respect to y direction

LS Least Square

LS_i Least Square using Intensity data

LS_ir Least Square using Intensity/Range data

LS_r Least Square using Range data

MAE Mean Absolute Error

NRMSE Normalized Root Mean Square Error

RGB-D Colours and depth images

RMSE Root Mean Square Error

SL Structured Light Camera

TLS Total Least Square

TLS_i Total Least Square using Intensity data

TLS_ir Total Least Square using Intensity/Range data

TLS_r Total Least Square using Range data

TOF Time-Of-Flight Camera

Z Depth image

Z_t Depth image derivative with respect to time

Z_x Depth image derivative with respect to x direction

Z_y Depth image derivative with respect to y direction

ψ Robust estimator function

ψ_d Robust estimator for data term

ψ_s Robust estimator for smoothness term

Chapter 1

Introduction

This chapter briefly introduces the thesis topic. The thesis is concerned with the computation of 3D range flow (3D sensor motion relative to 3D environmental points) and 2D optical flow. In order to fully understand the 3D range flow, we first need to define 2D and 3D optical flow. This chapter introduces these important definitions on optical flow by explaining the meaning of 2D optical flow, 3D optical flow, 3D scene flow, and 3D range flow. In addition, some of the application domains that use optical flow and range flow will be mentioned in the next section of this chapter. A summary of our thesis contributions, along with the outline of the coming chapters is provided at the end of this chapter.

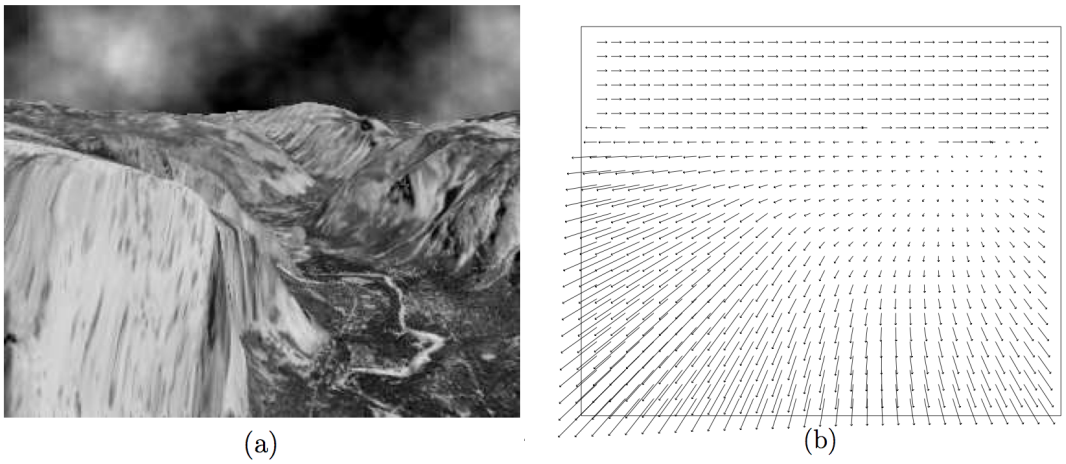
1.1 Optical Flow

Optical flow estimation is one of the established and most active research fields in computer vision and image processing. Starting from the original approaches by Horn and Schunck [2] and Lucas and Kanade [1], researchers have proposed many new methods to accurately measure optical flow.

Optical flow, also known as **image velocity**, is the apparent visual motion that you experience as you move through the world. For example, one perceives optical flow while sitting in a moving car and looking out the window–objects passing by will appear to move backward. The speed of this motion depends on the relative distance of these objects. Objects that are far away move much slower than objects which are closer, which move faster. Individual motions are called optical flow or image velocities. The entire motion field is called the optical flow field.

Optical flow methods try to calculate the motion between two image frames, which were acquired at times t and $t + \delta t$. Optical flow calculates the corresponding pixel motion induced on the image plane [11]. This shows the distribution of apparent velocities in the movement of brightness patterns in an image sequence [2]. In the case of a 2D image sequence, it specifies how much each pixel moves between adjacent images. For 3D images, the 3D optical flow specifies how much each voxel moves between adjacent volumes in a dataset. Figure 1.1 shows an example of a synthetic image (the 9^{th} image of the famous Yosemite fly through sequence)

and its correct 2D optical flow field. It shows vectors at each pixel specifying where that pixel will move to in the next image.



(a) (b)

Figure 1.1: (a) The middle frame from the Yosemite Fly-Through sequence and (b) its correct flow field.

Scene flow and **range flow** are particular types of 3D optical flow, in that they are both visible surface points in a 3D scene. Scene flow is computed from a stereo disparity map (and its gradient map) and the 2D optical flow of the left and right images of these stereo images. Conversely, range flow is computed solely from a depth map and its spatio-temporal derivatives. Both scene flow and range flow can be described as the 3D optical flow on visible environmental surfaces. Scene flow is both depth-disparity and intensity-based, while basic range flow is depth-based only.

1.2 Part of The Application Domains

Many computer vision systems depend on the information provided by optical flow in a variety of application domains. The importance of optical flow can be demonstrated through the following examples: motion estimation, video compression, object detection and tracking, image dominant plane extraction, movement detection, robot navigation, visual odometry, traffic analysis, and vehicle tracking [12].

Optical flow plays an important role in solving several video analysis problems. Action recognition is an essential task in the semantic interpretation of video content [13, 14, 15]. Video compression standards, like MPEG, exploit motion estimation to predict intermediate frames [16]. Both video indexing, video retrieval [17, 18, 19], and video restoration of old films [20, 21] are fields of interest that can be improved by taking flow estimation into account [12].

In a biomedical context, dynamic properties of tissues or cellular objects are fundamental. Optical flow is a required computation in several medical applications, such as the deformation of organs [22, 23] or the estimation of blood flow systems [24]. Medical image registration

and optical flow are linked by many conceptual and methodological links [25, 26, 27]. In microscopy, dense motion can inform cell deformation [28, 29], the motion of cellular structures [30, 29], and aid in individual cell tracking [31].

Input control systems for automatic guidance in robots or vehicle navigation also exploit optical flow computation. In real-world environments, researchers have demonstrated interest in exploiting optical flow in tasks such as autonomous car driving [32, 33, 34], obstacle detection, and obstacle avoidance [35, 36, 37]. In automated video surveillance, motion analysis is important for facial expression, gesture recognition [38, 39], crowd motion, and pedestrian behaviour analysis [40, 41, 12].

Depth sensors are now common and accessible in many popular consumer devices, which has increased scholarly attention on RGB-D scene flow estimation. Presently, range and depth information can be obtained not just with state-of-the-art technology and instrumentation owned by research labs or major companies, but also through tools such as consumer-grade depth cameras [42]. For example, the iPhone X uses range and depth information in their front camera to recognize faces for device unlocking. These examples demonstrate the broad range of data applications in many different domains. Indeed, motion analysis in a 3D range data is a critical research area with many different applications to explore.

Scene segmentation studies can be assisted by depth data in three main applications. They are: scene matting with colour and depth data for a single frame [43, 44, 45, 46] or video [47, 48, 49, 50], scene segmentation [51, 52, 53, 54], or semantic segmentation [55, 56, 57]. Human pose estimation, tracking [58, 59, 60], and gesture recognition are [61, 62, 63, 64] also enhanced by depth data provided from consumer cameras.

Many studies have analyzed range flow, largely because there are so many potential applications of range sensors and range data. Range flow estimation is an extremely important problem in a growing field. Range flow helps to estimate the 3D elastic constants of objects [65]. Consequently, range flow estimation can be used to compute surface expansion rates to study 3D leaf motion and plant growth [66, 67, 68, 69] and also to predict severe weather storm displacement by tracking deformable objects automatically [70]. Landslide displacement monitoring also requires multi-temporal airborne and terrestrial laser scanning for accurate range estimation [71, 72]. In [73], a visual odometry algorithm was based on geometric data in a robotics range sensor that required a 3D range flow estimation algorithm. In [74, 75, 76], estimation was calculated in an odometry planar motion with a laser scanner by applying the range flow constraint equation for each scan. Similarly, Ismaeil et al. [77] enhanced depth videos with non-rigid deformations by using 3D range flow in their framework.

In biomedical research, the 3D-MATIC Research Laboratory exploits range flow estimation techniques to study deformations of the human body [78]. This helps show the organs' motion, such as in the medical imaging study that used MRI cardiac datasets to show the range flow for a beating heart [79].

Facial analysis and synthesis schemes estimate the static and dynamical parameters that, respectively, correspond to the structure and motion of the face. It is required to estimate 3D face movements and non-rigid facial expressions, and also for extracting MPEG-4 facial animation

parameters [80, 81].

3D range flow plays a vital role in advanced vehicle-based safety and warning systems that use laser scanners to measure road geometry. This requires high-resolution images that can accurately perform object tracking and velocity estimation [82, 83, 84]. Tracking objects in a night vision system with range gated cameras also requires 3D range flow [85]. Similarly, ego-motion in vehicles is estimated using visual odometry methods that rely on the range constraint equation[86]. This is consistent with Menze's recent study [87], in which they applied scene flow estimation in autonomous driving systems.

1.3 Thesis Contributions

This thesis focuses on 3D range flow using RGB-D data. It is indebted to the valuable work on range flow as summarized by Spies and Barron [3]. The following list briefly discusses our thesis contributions.

1. We are re-implementing 3D range flow, as proposed by Spies and Barron [3], in using more modern approaches such as MATLAB language. Our framework extended four classic optical flow algorithms to handle 3D range flow. One contribution is casting the 3D range flow to a hierarchical framework. This allowed us to handle faster 3D motions that may be aliased (under-sampled in time). We are using coarse to fine warping framework, and we are introducing a primitive warping technique for Z pyramid to remove dw before projecting the Z values to the next level.

2. We recasted a local approach to estimate 2D optical flow, consistent with the approach of Lucas and Kanade [1], using intensity data in our hierarchical framework. Similarly, we estimated 3D range flow using local range data as proposed in [88]. For this local approach, we used gradient constraints from nearby pixels, assuming they shared the same 2D velocity. We implemented two versions: one with total least-squares and the second with ordinary least-squares.

3. We also considered Horn and Schunck's **(HS)** approach for global regularization [2]. We implemented (HS) framework to estimate 2D optical flow with two other approaches using range data to regularize 3D range flow [89] and recast them in our hierarchical framework.

4. We also implemented an indirect regularization optical and range flow that uses the 2D and 3D least square results as the initial values for the regularization. Note that direct regularization did not use any initial values from the least squares. While it will produce the same results, it does require more iterations (and time) to do so.

5. We also recasted Bruhn et al.'s [5] model for 2D optical flow in our hierarchical scheme for range data and compute 3D range flow. Notably, Bruhn et al. had previously combined the important advantages of the local and global approaches. They observed that the smoothing steps in each of these methods serve different purposes with differing advantages and shortcomings. Bruhn's method yields dense flow fields (as in the global scheme) that are robust against noise (as in the local scheme).

6. Finally, we recasted Brox et al.'s [6] optical flow scheme to work with range data. The Brox et al. method combines three assumptions: a brightness constancy assumption, a gradient constancy assumption, and a discontinuity-preserving spatio-temporal smoothness constraint. A numerical scheme based on fixed point iterations implements a coarse-to-fine warping strategy, which gives a theoretical foundation that allows for the computation of larger optical flow vectors. We used the same Brox et al. constraints for range data.

7. Our research used RGB-D datasets, which contain intensity values as well as depth information for each pixel. Therefore, we used intensity and range data to estimate 3D range flow. This process enhances the algorithms and yields more accurate range flow, especially when compared with range flow computations from range data alone or optical flow computed from intensity data alone. All proposed algorithms listed above were extended to estimate 3D range flow using both range data and intensity data together.

8. We used a robust statistical formulation as a penalty function in the algorithms to compute optical and range flow. To reduce the number of outliers in the output optical flow and to overcome the discontinuity problem in the data, robust estimation is necessary. Brox et al. [6] applied the Charbonnier function in a form: $\Psi(s^2) = \sqrt{s^2 + \epsilon^2}$ to increase concave function. Alternately, Sun [90] investigated several penalty functions such as Lorentzian, Charbonnier, Generalized Charbonnier, and Geman Mcclure methods. We recasted our local and global approaches to different robust estimations in both 2D optical flow and 3D range flow.

9. We studied four types sensors and cameras that provide RGB-D data. Using this kind of data, we computed optical flow and range flow. These were inexpensive range sensors and cameras that use different technologies to extract depth information for 3D range datasets. The Kinect V2 recovers depth information using Time-of-Flight (TOF) technology, while the ZED camera utilizes the stereo vision technique. The iPhone X has two cameras, front and rear: the front camera receives depth information using a structured light (SL) technique; the rear camera uses stereo vision method. In Professor Barron's lab at Western University, we used linear and rotation positioners to acquire range sequences with known 3D motion sensor and 3D scene depth maps. We captured the scene images while moving the cameras with translation and diverging motions. We subsequently analyzed depth data for the same scenes using four different cameras. Receiving information from a number of range sensors that generated the same range data enabled us to compare the accuracy of the depth maps as gathered by different devices.

10. We prepared datasets from acquired images using Kinect V2, ZED, and iPhone X. Each required several pre-processing steps in order to generate 2D optical flow and 3D range flow for the same symmetrical scene. Moreover, we used a new method to calibrate and align the colour and depth channels in Kinect V2 images using a registration function in MATLAB.

11. We applied the newly generated 3D range datasets to our algorithms in the implemented framework to evaluate the capacity of different depth maps to measure 3D range flow.

12. We evaluated and tested the proposed algorithms using the acquired range datasets on an old NSERC range sequence (1997-1999). In addition, we provide a quantitative and qualitative analysis of the algorithms using the translating and diverging motions included in the generated datasets. Several error measurements were applied to provide the quantitative analysis such as AAE (Average Angular Error), RMSE (Root Mean Squared Error). Overall, this thesis provides a wide range of computational tools for computing range flow and quantitatively and qualitatively analyzing the results.

13. We compared our approaches with other scene and range estimation flow studies using Middlebury 2002 datasets [91], *Teddy*, and *Cones* sequences.

14. Finally, we displayed correct and computed 3D range flows as vector fields rather than colour images. We preferred vector fields because colour representation often hides problems in the flow fields which are very visible in the vector fields. For 3D flow, we showed the 3D range flow in two simple ways: (u,v) in one 2D plot and (u,w) in another 2D plot, or as combined into a single 3D plot (u,v,w).

1.4 Thesis Overview

- Chapter 1: Briefly defines terms related to optical flow, including 2D, 3D, scene, and range flow. We list thesis contributions and provide chapter outlines.

- Chapter 2: A literature survey of research on 2D and 3D optical flow techniques to-date. We also discuss various scene and range flow work as it pertains to our study.

- Chapter 3: Explanation of how we implemented optical flow and range flow algorithms in full mathematical details.

- Chapter 4: Further elaboration on the framework of our implementation.

- Chapter 5: Discussion of the 3D imaging techniques and the four sensors/cameras that were used to acquire range data.

- Chapter 6: Explanation of the datasets used to evaluate and test the algorithms. In addition, we display the correct optical flow and range flow for each sequence.

- Chapter 7: Discusion of the results and a quantitative and qualitative analysis of the algorithms.

- Chapter 8: Summarization of the whole thesis and concluding thoughts on our work. In addition, we provide suggestions for further research and consideration.

Chapter 2

Literature Review

This chapter provides a literature review on relevant optical flow algorithms as they relate to this thesis. The first section discusses the topic background. Specifically, it describes basic concepts for 2D and 3D optical flow and introduces pertinent scene and range flow algorithms. The second section of this chapter discusses related research, including prior studies on scene and range flow.

2.1 Background

2.1.1 2D Optical Flow

2D optical flow methods calculate the motion or displacement of pixels between two image frames acquired at times t and $t + \delta t$. It assumes that the scene lighting is Lambertian, which finds the following: intensity variations between adjacent images are caused by the motion of the camera with respect to the scene rather than other artifacts such as changing illumination or deforming objects; that local motion can be well approximated by pure translation; and that image intensity derivatives are due to motion only. In as much as these assumptions are satisfied, optical flow can be approximated in the local 2D motion in image sequences.

The 2D Motion Constraint Equation

The initial concept of a 2D motion constraint equation originates from the brightness constancy assumption in Horn and Schunck's 1981 pioneering method of calculating the optical flow algorithm [2]. A basic assumption in this algorithm is that if a pixel moves from one location to another, the grayvalues or intensities in the neighbourhood of that pixel remains constant. In other words, even if an object changes position during the short time interval from t_1 to t_2, the reflectivity and illumination of the object will remain the same. Mathematically, this can be expressed as:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{2.1}$$

Where $I(x, y, t)$ is the intensity of the image at position (x, y) and time t, while δx, δy is the change in position of that pixel over time δt. If we perform a 1st order Taylor series expansion

about $I(x, y, t)$, we get

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + H.O.T. \tag{2.2}$$

We ignored the H.O.T. and combined Equation (2.1) and Equation (2.2) to obtain the motion constraint equation:

$$I_x u + I_y v + I_t = 0 \tag{2.3}$$

$$\nabla I \cdot \vec{v} + I_t = 0 \tag{2.4}$$

Here $u = \frac{\delta x}{\delta t}$ and $v = \frac{\delta y}{\delta t}$ are the x and y components of image velocity or optical flow. $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are image intensity derivatives at $I(x, y, t)$ and $\vec{v} = (u, v)$. In Equation (2.4), $\nabla I = (I_x, I_y)$ is the spatial intensity gradient and $\vec{v} = (u, v)$ is the optical flow at pixel (x, y) at time t. $\nabla I \cdot \vec{v} = -I_t$ is called the *2D Motion Constraint Equation*.

2D Aperture Problem

The 2D motion constraint equation is a line equation that has one equation with two unknowns. This problem is called the 2D aperture problem and needs to be solved when computing 2D optical flow. There is usually deficient local image intensity structure to measure full optical flow, but enough structure to measure the normal to the local intensity structure flow [92]. This means that only the component of the flow perpendicular to the local intensity structure (in the direction of the intensity gradient) can be computed, while the tangential component of the flow cannot be computed. In order to compute full velocity or full optical flow, additional constraints must be incorporated into the solution.

To overcome and solve this problem, full optical flow computation requires additional constraints. Lucas and Kanade [1], Horn and Schunck [2], and Brox et al. [6] all used different methods to overcome the aperture problem. The next section gives abbreviated descriptions about these algorithms.

2D Lucas and Kanade

Lucas and Kanade's technique [1] is a pioneering algorithm in optical flow area. This work was originally written for registration in stereo images, which is equivalent to the optical flow problem. It is based on two key assumptions. The first assumption is the grayvalues constancy assumption, which leads to the use of the motion constraint equation as seen in Equation (2.3). The second assumption of this algorithm is that the velocity is constant in some local neighbourhoods (the local constant velocity model); consequently, two (or more) sets of derivatives can yield a non-singular linear (the least squares) system of equations that yield for values (u, v).

Barron et al. [92] implemented a weighted least-squares (LS) fit to local first-order constraints to a constant model $\vec{v}$ in each small spatial neighbourhood Ω by minimizing:

$$\sum_{x,y \in \Omega} [\Delta I(x, y, t).\vec{v} + I_t(x, y, t)]^2, \tag{2.5}$$

The solution to Equation (2.5) is given as:

$$\vec{v} = [A^T A]^{-1} A^T \vec{b}, \tag{2.6}$$

where for N pixels (for a $n \times n$ neighbourhoods $N = n^2$), $(x_i, y_i) \in \Omega$ at a single time t:

$$A = [\Delta I(x\_1, y\_1), ..., \Delta I(x\_N, y\_N)],$$
$$\vec{b} = -(I\_t(x\_1, y\_1), ..., I\_t(x\_N, y\_N))$$

The solution to Equation (2.6) can be solved in closed form when $A^T A$ is a non-singular matrix. $A^T A$ is the 2×2 matrix:

$$A^T W^2 A = \begin{bmatrix} \sum I_x^2(x,y) & \sum I_x(x,y) I_y(x,y) \\ \sum I_y(x,y) I_x(x,y) & \sum I_y^2(x,y) \end{bmatrix} \tag{2.7}$$

where all sums are taken over points in the neighbourhood Ω. Spatial neighbourhoods Ω of 5×5 pixels worked well.

2D Horn and Schunck

The Horn and Schunck algorithm [2] is another pioneering technique in the optical flow computation area. It successfully estimates the optical flow between two 2D images and is based on two key assumptions: the grayvalue constancy assumption and the global smoothness assumption. The grayvalue constancy assumption assumes that even if a pixel moves from one location to another, the grayvalues of the pixel remain constant. So $\frac{\partial I}{\partial t} = 0$, which can be expanded using a 1st order Taylor series expansion to derive the motion constraint equation as seen in Equation (2.3).

The global smoothness assumption requires neighbouring points on the objects to have similar velocities and the optical flow of the brightness patterns in the image to vary smoothly almost everywhere. This additional smoothness constraint minimizes the square of the magnitude of the optical flow velocity's gradient:

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{2.8}$$

The problem then becomes minimizing the total energy function:

$$\int\int (I_x u + I_y v + I_t)^2 + \alpha^2 \left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right] dxdy \tag{2.9}$$

where α^2 is a weighting factor between the brightness constraint and smoothness constraint; larger values mean more smoothing in optical flow.

The Euler-Lagrange derivatives of Equation (2.9) can be minimized using the Gauss-Seidel method that iteratively solves it to produce a globally smooth optical flow field. The minimization of this function will yield the optimal optical flow velocity (u, v). It would be very costly and likely impossible to solve these linear equations simultaneously by one of the standard methods. For example, a 100×100 image require the solutions of a square linear system

of 10,000 equations, which is not visible because of round-off error. Therefore, Horn and Schunck compute a new set of velocity estimates (u^{n+1}, v^{n+1}) from the estimated derivatives and the average of the previous velocity estimates (u^n, v^n) as:

$$u^{n+1} = \bar{u}^n - I_x[I\_x\bar{u}^n + I\_y\bar{v}^n + I\_t]/(\alpha^2 + I\_x^2 + I_y^2)$$
$$v^{n+1} = \bar{v}^n - I_x[I\_x\bar{u}^n + I\_y\bar{v}^n + I\_t]/(\alpha^2 + I\_x^2 + I\_y^2)$$

The averages of u and v are usually initialized to 0 for each pixel. Eventually, the difference in the optical flow between two successive iterations will become very small, at which point we assume that we have reached the optimal solution.

Both Lucas and Kanade's algorithm and Horn and Schunck's techniques compute optical flow for small motions and cannot deal with large displacement. We solved this problem by using a hierarchical approach with image warping and non-linearized model equations, as described below. Of course, the hierarchical framework can also be added to Horn and Schunck's and Lucas and Kanade's optical flow algorithms.

2D Bruhn et.al

Differential optical flow can be classified into local methods, such as Lucas and Kanade's method, [1] or global methods, such as Horn and Schunck's global regularization [2]. Often local methods are more robust under noise, while global techniques yield dense flow fields. As a result, Bruhn et al. [4, 5] proposed a method that combined the advantage of local and global optical flow methods.

Horn and Schunck determine optical flow (u, v) by minimizing the global energy functional:

$$E_{HS}(u, v) = \int_{\Omega(x,y)} ((I_x u + I_y v + I_t)^2 + \alpha(|\nabla u|^2 + |\nabla v|^2))dxdy, \tag{2.10}$$

where $\alpha > 0$ is the regularization term weighting the importance of the motion constraint equation and the smoothness term and $\Omega(x, y)$ is the image.

Lucas and Kanade assume u and v are constant in some local neighbourhood of size ρ, then (u, v) are found by minimizing:

$$E_{LK}(u, v) = K_\rho \left((I_x u + I_y v + I_t)^2\right). \tag{2.11}$$

Here K_ρ is the standard deviation of the Gaussian used to smooth the images.

Bruhn et al. [5] designed a hybrid method that combines both local and global constraints (CLG). Using the definitions:

$$\mathbf{w} = (u, v, 1)^T, \tag{2.12}$$
$$|\nabla \mathbf{w}|^2 = |\nabla u|^2 + |\nabla v|^2, \tag{2.13}$$
$$\nabla_3 I = (I_x, I_y, I_t)^T \text{ and} \tag{2.14}$$
$$J_\rho(\nabla_3 I) = K_\rho(\nabla_3 I \nabla_3 I^T) \tag{2.15}$$

Lucas and Kanade minimize:

$$E_{LK}(\mathbf{w}) = \mathbf{w}^T J_\rho \nabla_3 I \mathbf{w} \tag{2.16}$$

while Horn and Schunck minimize:

$$E_{HS}(\mathbf{w}) = \int_{\Omega(x,y)} (\mathbf{w}^T J_0(\nabla_3 I)\mathbf{w} + \alpha(|\nabla \mathbf{w}|^2) dxdy, \tag{2.17}$$

where J_0 is $(\nabla_3 I \nabla_3 I^T)$, we replace J_0 with $J_\rho(\nabla_3 I)$ and minimize:

$$E_{3CLG}(\mathbf{w}) = \int_{\Omega(x,y,t)} (\mathbf{w}^T J_\rho(\nabla_3 I)\mathbf{w} + \alpha(|\nabla \mathbf{w}|^2) dxdydt. \tag{2.18}$$

Here $\Omega(x, y, t)$ is the image sequence, Gaussian convolution with standard deviation ρ is done in time and

$$|\nabla_3 \mathbf{w}|^2 = |\nabla_3 u|^2 + |\nabla_3 v|^2. \tag{2.19}$$

Robust methods can be used to suppress outliers, such as Bruhn et al.'s technique which uses Charbonnier et al.'s [93] method. The new minimizer as follows:

$$E_{3CLG-N}(\mathbf{w}) = \int_{\Omega(x,y,t)} \left(\psi_1(\mathbf{w}^T J_\rho(\nabla_3 I)\mathbf{w} + \alpha\psi_2(|\nabla_3 \mathbf{w}|^2)\right) dxdydt. \tag{2.20}$$

where:

$$\psi_i(s^2) = 2\beta_i^2 \sqrt{1 + \frac{s^2}{\beta_i^3}}, \quad i \in 1, 2 \tag{2.21}$$

and β_1 and β_2 are scaling parameters. They proposed a multi-resolution (hierarchical) method where the computed optical flow field at one level is used to warp (correct) the original sequence before moving to the next finer level (the computed (u, v) is not used as the initialization for the optical flow calculation at the next finer level).

2D Brox et.al

Brox et al. [6] give among the best optical flow results in terms of accuracy. To compute optical flow, they proposed an energy functional that combines three assumptions: brightness constancy, gradient constancy, and a discontinuity-preserving spatio-temporal smoothness constraint. In addition, they applied pyramid coarse to fine warping technique.

The grayvalue and gradient assumptions are measured by the energy:

$$\begin{aligned} E_{Data}(u, v) = \int_\Omega \Psi(|I(x + u, y + v, t + 1) - I(x, y, t)|^2) \\ +\gamma(|\nabla I(x + u, y + v, t + 1) - \nabla I(x, y, t)|^2) dxdydt. \end{aligned} \tag{2.22}$$

The smoothness term that models the assumption of piecewise smoothness is:

$$E_{smooth}(u, v) = \int_\Omega \Psi(|\nabla_3 u|^2 + |\nabla_3 v|^2) dxdydt. \tag{2.23}$$

Here $\nabla_3 = (\partial_x, \partial_y, \partial_t)^T$ is the spatio-temporal gradient. The total energy is:

$$E(u, v) = E_{Data} + \alpha E_{smooth}. \tag{2.24}$$

According to the calculus of variations, a minimizer of the equation must satisfy the Euler-Lagrange equations that yield to:

$$\Psi' \left(I_z^2 + \gamma(I_{xz}^2 + I_{yz}^2)(I_x I_z + \gamma(I_{xx} I_{xz} + I_{xy} I_{yz})) - \alpha \mathbf{Div} \Psi'(|\nabla_3 u|^2 + |\nabla_3 v|^2) \nabla_3 u \right) = 0 \tag{2.25}$$

and

$$\Psi' \left(I_z^2 + \gamma(I_{xz}^2 + I_{yz}^2)(I_y I_z + \gamma(I_{yy} I_{yz} + I_{xy} I_{xz})) - \alpha \mathbf{Div} \Psi'(|\nabla_3 u|^2 + |\nabla_3 v|^2) \nabla_3 v \right) = 0 \tag{2.26}$$

We used a consistent numerical scheme based on two fixed point iterations combined with a downsampling strategy to solve the Euler-Lagrange equations. In order to find the global minimum, we used a multiscale warping strategy. First, we solved a coarse, smoothed version of the problem. Then, the coarse solution is used to warp the second image back into the first. This is done by building an image pyramid with a certain number of levels and applying a downsampling factor $\eta = 0.5$ from the original image. The computation of velocities begins from the coarsest level. Once a solution is obtained, the velocities are projected down to the adjacent finer level, and are used to warp the second image back to the first one. The construction of the image pyramid and the projection of velocities between adjacent levels are performed by bilinear interpolation. If the warping at a level was well performed and the coarse optical flow was good on the previous level, most of the motion between the two images will have been removed. Further optical flow calculations and warping as one descends the pyramid can lead to a good final optical flow, even for long image motions.

Chapter 3 further discusses the mathematical and theoretical details that relate to our implementation of these algorithms in more detail.

2.1.2 3D Optical Flow

3D optical flow was computed for 3D images. 3D optical flow specifies how much each voxel moves between adjacent volumes in a dataset. 3D optical flow is a simple extension of 2D optical flow. Therefore, 3D optical flow methods calculate the motion or displacement of voxel between two images acquired at times t and $t + \delta t$.

The 3D Motion Constraint Equation

The 3D motion constraint equation is a simple extension of the 2D motion constraint equation [92]. Since $I(x, y, z, t) = I(x + \delta x, y + \delta y, z + \delta z, t + \delta t)$, we can perform a 1st order Taylor series expansion (as in the 2D case) and obtain:

$$I_x u + I_y v + I_z w + I_t = 0 \tag{2.27}$$

3D Aperture Problem

3D Motion Constraint Equation is a description of plane motion in 3D space. Any point on the plane(s) is possibly the correct 3D velocity. The aperture problem in 3D yields two types of normal velocity: plane and line [92].

The velocity on the plane closest to the origin is called the normal plane velocity. The normal velocity line is the velocity caused by the intersection of two planes closest to the origin. Finally, the single point at which three planes intersect is the full 3D velocity. Graphical representation of the flow types (full, line, and plane) are shown in Figure 2.1 [92].

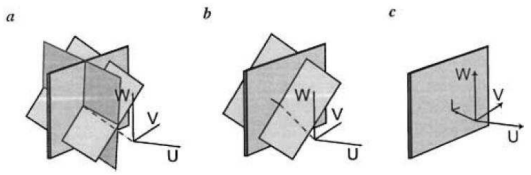


Figure 2.1: Types of 3D optical flows: (a) full flow, (b) line normal flow, and (c) plane normal flow. [7]

In the next section, we describe the modification of two classic approaches to calculate 3D optical flow. First, we discuss the seminal work of Horn and Schunck [2] and Lucas and Kanade [1] that have been done by Barron [92]. We then discuss how Barron and Chen [94] extended Brox et al. [6] into 3D.

3D Lucas and Kanade

Barron [92] used an extension of the 2D Lucas and Kanade technique to compute volumetric optical flow in MRI data. Using the 3D motion constraint Equation (2.27), it can be solved by setting up a system of linear equations:

$$\vec{V} = [A^T A]^{-1} A^T B, \tag{2.28}$$

The N rows of consist of A consist of I_x, I_y, I_z for each (x, y, z) position and the N rows of B consist of the $-I_t$ values for those (x, y, z) that is:

$$A = [\Delta I(x_1, y_1, z_1), ..., \Delta I(x_N, y_N, z_N)],$$
$$\vec{B} = -(I_t(x_1, y_1, z_1), ..., I_t(x_N, y_N, z_N))$$

3D Horn and Schunck

Barron also extended the 2D Horn and Schunck technique to compute 3D volumetric optical flow [92]. Using the 3D motion constraint Equation (2.27), we determine the Horn and

Schunck regularization term as:

$$\sum_{R}(I_x u + I_y v + I_z w + I_t) + \alpha^2[(\frac{\partial u}{\partial x})^2 + (\frac{\partial u}{\partial y})^2 + (\frac{\partial u}{\partial z})^2 + (\frac{\partial v}{\partial x})^2 + (\frac{\partial v}{\partial y})^2 + (\frac{\partial v}{\partial z})^2 + (\frac{\partial w}{\partial x})^2 + (\frac{\partial w}{\partial y})^2 + (\frac{\partial w}{\partial z})^2] \tag{2.29}$$

Iterative Gauss-Seidel equations that minimize the Euler-Lagrange equations are based on this integral:

$$u^{k+1} = \bar{u}^k - \frac{I_x[I_x\bar{u} + I_y\bar{v} + I_z\bar{w} + I_t]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)}$$
$$v^{k+1} = \bar{v}^k - \frac{I_y[I_x\bar{u} + I_y\bar{v} + I_z\bar{w} + I_t]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)}$$
$$w^{k+1} = \bar{w}^k - \frac{I_z[I_x\bar{u} + I_y\bar{v} + I_z\bar{w} + I_t]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)}$$

where $\bar{u}^k, \bar{v}^k, \bar{w}^k$ are $n \times n \times n$ averages of neighbourhoods of velocities at iteration k ($n = 5$ gives good results).

3D Brox et.al

Chen and Barron extended Brox et al.'s 2D algorithm 2.1.1 into 3D [94]. The data term consisting of brightness constancy and gradient constancy assumptions became:

$$E_{Data}(u, v, w) = \int_{\Omega} \Psi(|I(x + u, y + v, z + w, t + 1) - I(x, y, z, t)|^2) + \gamma(|\nabla I(x + u, y + v, z + w + t + 1) - \nabla I(x, y, z, t)|^2) dxdydzdt. \tag{2.30}$$

Here Ψ does robust estimation by ensuring the function is convex (a single global solution results) and $\Psi(s^2) = \sqrt{s^2 + \epsilon}$. Brox et al. use $\epsilon = 0.001$. A smoothness term that models the assumption of piecewise smoothness is:

$$E_{smooth}(u, v, w) = \int_{\Omega} \Psi(|\nabla_3 u|^2 + |\nabla_3 v|^2 + |\nabla_3 w|^2) dxdydzdt. \tag{2.31}$$

Here $\nabla = (\partial_x, \partial_y, \partial_z)$ is the spatio-temporal gradient. The total energy is the weighted sum between the data term and the smoothness term:

$$E(u, v, w) = E_{Data} + \alpha E_{smooth}. \tag{2.32}$$

Specific Types of 3D Optical flow

Range and **scene** flow are the 3D motion of pixels on a moving surface relative to the sensor. This is contrary to 3D optical flow, which is the 3D motion of each voxel in a volume sequence. To compute 3D optical flow, we need 3D data such as MRI data. In contrast, for scene and range

flow, we are computing the 3D optical flow using 2D data (intensity and depth) to observe the motion of the objects' surface.

Scene and range flow are distinguished by how they are computed. Scene flow requires disparity and disparity gradients maps computed from a stereo sequence and the 2D optical flow of the left and right images to compute 3D motion. Consequently, scene flow does not use any 3D information directly for input. On the other hand, range flow uses a depth map of a scene measured over time. These maps can be measured using a range scanner, or computed from the disparity map recovered from stereo images or other methods that can provide depth information. Range flow uses range flow motion constraint equation.

2.1.3 Range Flow

Range flow is the instantaneous 3D-velocity field of a moving surface that can be recovered from range datasets. Range flow uses a 3D depth map of a scene measured over time. These maps can be measured by a range scanner, computed from the disparity map recovered from stereo images, or computed from relative depth maps recovered from a monocular motion and structure algorithm. [7]. In the next section, we discuss several important concepts on range flow (the range flow motion constraint equation and the 3D aperture problem) as mentioned in Spies and Barron's work from 1999 to 2002.

Range Flow Motion Constraint Equation

The range flow motion constraint equation was first introduced by Horn and Harris [95] in the context of range scanners without intensity information.

The observed surface can be described by its depth as a function of space and time $Z = Z(X, Y, t)$. The derivative of Z with respect to time yields the range flow motion constraint equation [96]:

$$Z_X u + Z_Y v + w + Z_t = 0 \tag{2.33}$$

Here $f = [u, v, w]^T$ is the 3D range flow, and indices denote partial derivatives and Z_X, Z_Y and Z_t are spatio-temporal derivatives of the depth coordinate Z. The range flow motion constraint equation is the analog to the brightness change constraint equation used for optical flow. As this gives only one constraint equation in three unknowns, we need to make further assumptions; this is the aperture problem revisited [7].

Range Flow Aperture Problem

Equation (2.33) poses one constraint in three unknowns, which may be attributed to our initial assumption of locally planar surface patches. On a plane surface, only the movement perpendicular to the plane may be observed, which is the aperture problem revisited. Equation(2.33) describes a plane in $(u, v, w) - space$ with the surface normal $[Z_x Z_y 1]^T$. The best solution under these circumstances will be the minimal vector between the origin and the constraint plane.

This gives the *raw normal flow*:

$$f_r = \frac{-Z_t}{Z_X^2 + Z_Y^2 + 1} \begin{bmatrix} Z_X \\ Z_Y \\ 1 \end{bmatrix} \tag{2.34}$$

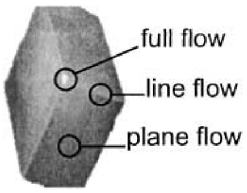


Figure 2.2: Example range data from a cube that illustrates the three different types of neighbourhoods encountered in three-dimensional data [7].

The three possible types of neighborhoods are illustrated in Figure 2.2. All three motion components can be determined locally; this flow is the *full flow*. Any movement on the edges or linear structures cannot be resolved locally–only the velocity in the perpendicular plane can be determined, which is called this *line flow*. On planar surfaces, any velocity within the plane cannot be estimated; only the component perpendicular to the surface can be estimated. This type is the *plane flow* [7]. The three types of flows are shown in Figure 2.1.

Spies and Barron papers from 1999 to 2002

In the next section, we provide a brief description of the range flow approaches that Dr. John Barron completed with other authors. As previously stated, we provide more mathematical details about these algorithms in Chapter 3.

Total Least Square

Spies et al. in [88] computed the local 3D-motion of the object in the scene (range flow) using a given sequence of depth map data gathered by a Biris laser range sensor. Using a range flow constraint equation, they solved the 3D aperture problem by using a total least squares method. Then, they computed range flow (full, line, and plane) using eigenvalues and eigenvectors.

This method identifies areas where line, plane normal, and full flow can be computed. Their performance was assessed on both synthetic and real datasets.

Indirect Regularization

Spies et al. [89] extended the differential total least squares method for range flow estimation in [88] to a global approach. However, to reduce the computational cost, they only computed the range flow where the trace of the tensor matrix is greater than a certain threshold. This can lead to holes in the depth data. Consequently, they introduced an iterative regularization approach

to fill in these holes and compute dense range flow. A simple iterative regularization algorithm computes varying flow fields smoothly in the previously segmented area by total least square (TLS).

They estimated a dense and smooth flow field $v = \begin{bmatrix} U & V & W \end{bmatrix}^T$. In places where flow estimations from the above TLS algorithm exist, they denote them f. Each estimated flow vector $f_{f,p,l}$ constrains the solution within this subspace. Therefore, they require the regularized solution to be close in the total least squares sense.

$$(Pv - f)^2 \rightarrow min \tag{2.35}$$

At locations where no solution has been computed, it is clear that no such data term exists. To ensure smoothly varying parameters, they use a smoothness term:

$$\sum_{i=1}^{3} (\triangledown v_i)^2 \rightarrow min \tag{2.36}$$

Combining the data in Equation (2.35) and smoothness in Equation (2.36) in the considered area A yields the following minimization problem.

$$\int_A \left\{ \omega (Pv - f)^2 + \alpha \sum_{i=1}^{3} (\nabla (v_i)^2 \right\} dr \rightarrow min \tag{2.37}$$

In Equation (2.37) above, P is the projection matrix which projects onto the subspace determined by the TLS algorithm, ω is the confidence measure of the TLS algorithm, f is the combined plane flow, line flow, and full flow, $v = \begin{bmatrix} U & V & W \end{bmatrix}^T$ and $\nabla = [\partial x, \partial y, \partial t]^T$, which considers spatial neighbourhoods and enforces temporal smoothness. The confidence measure is used to detect motion discontinuities caused by sharp edges or occlusions and to remove noise because the TLS algorithm fails when there are motion discontinuities.

The minimum of Equation (2.37) is reached by the Euler-Lagrange equations. The derivatives were subsequently applied to the formula to get the iterative solution as follows:

$$v^{k+1} = \alpha A^{-1} v^{-k} + \omega A^{-1} P f \tag{2.38}$$

Direct Regularization

Instead of first performing a TLS analysis, we might tried to find the flow field by imposing the smoothness constraint [89]. The minimization is as follows:

$$\int_A \left\{ (d^T \cdot u)^2 + \alpha \sum_{i=1}^{3} (\nabla (v_i)^2 \right\} dr \rightarrow min. \tag{2.39}$$

Again, the minimum of Equation (2.39) is reached by the Euler-Lagrange equations. The derivatives were then applied to the formula to get the iterative solution as follows:

$$v^{k+1} = \alpha A_1^{-1} p^{-k} + d_4 A_1^{-1} d' \tag{2.40}$$

Spies et al. showed how the sparse information from a TLS based technique could be combined to yield dense full flow fields throughout an entire selected area. The segmentation into regions as it corresponds to different motions can easily be calculated based on the quality of the initial TLS estimation. The performance is quantitatively assessed on synthetic and real data and the algorithm is found to give good results. Finally, they show the motion of a living castor oil leaf.

Combined Intensity and Depth Data

In [97], Spies et al., they showed that combining intensity and depth information greatly helps in the estimation of the local 3D range flow of moving surfaces. They demonstrated how intensity and depth information could be combined in both a local total least squares algorithm and in an iterative global variational technique. They used two constraint equations; first, the range flow motion constraint equation:

$$Z_x u + Z_y v + w + Z_t = 0 \tag{2.41}$$

and, secondly, the brightness change constraint equation:

$$I_x u + I_y v + I_t = 0. \tag{2.42}$$

The next subsection describes this combination for both TLS and regularization methods.

Total Least Squares:

The range flow motion constraint given by Equation (2.41) and the bright change constraint given by Equation (2.42) are recast in a total least squares framework as follows:

$$u_1 Z_X + u_2 Z_Y + u_3 + u_4 Z_t = 0 \text{ and} \tag{2.43}$$

$$u_1 I_X + u_2 I_Y + u_4 I_t = 0.. \tag{2.44}$$

Here u_1, u_2, u_3 are the 3D velocities scaled by u_4 as computed by total least squares. The range flow is given by Equation (2.45) below:

$$\begin{bmatrix} U & V & W \end{bmatrix}^T = \frac{1}{u_4} \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^T. \tag{2.45}$$

The vector:

$$\vec{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T \tag{2.46}$$

is normalized and the following energy functional is minimized:

$$J = \int_A [(u_1 Z_X + u_2 Z_Y + u_3 + u_4 Z_t)^2 + \beta^2 (u_1 I_X + u_2 I_Y + u_4 I_t)^2 + \lambda(\vec{u}^T \vec{u})] dX dY \tag{2.47}$$

Here, λ is an eigenvalue, which is computed using convolution and smoothing operator. β^2 is a weighting term, weighing the relative importance of the intensity and depth terms.

Regularization:

To solve for the range flow using the global smoothness method, the following energy functional is minimized:

$$J = \int_A [(u_1 Z_X + u_2 Z_Y + u_3 + u_4 Z_t)^2 + \beta^2 (u_1 I_X + u_2 I_Y + u_4 I_t)^2 + \alpha^2 (\nabla U^2 + \nabla V^2 + \nabla W^2] dX dY. \quad (2.48)$$

Here, the smoothness term is weighed by the constant factor α^2, which specifies how important smoothness is in the solution.

In Equation (2.48), the intensity values for the data that Spies et al. used are about 100 times larger than the depth values, so the intensity values will dominate the numerical calculation (unless appropriately weighted). Therefore, the two constraints are weighted by the factor β^2, where β^2 is $\frac{<\|\nabla Z\|^2>}{<\|\nabla I\|^2>}$. Spies et al.'s results showed that the integrated use of intensity and depth data greatly improve range flow estimation.

Quantitative Studies

Barron and Spies [98, 99] showed quantitative results for computing total least squares and global regularized range flow on a real range sequence (NSERC datasets). There is a more complete description of this dataset in Chapter 6. In [99], they used least square (LS). This quantitative study was done solely with range data using a combination of image and range data. The difference between the true 3D velocity and the computation flow was computed. The NSERC sequence is perhaps the most difficult type of range sequence to analyze. Most of the surfaces are planar with little or no full or line normal velocity. The direct and indirect regularization algorithms were only able to compute full flow in the vicinity of this full and line normal flow. Consequently, the combined regularization approach uses both intensity and range data to obtain full flow everywhere. The usefulness of combining the two types of data is significant; this enables a better calculation of the flow than with the range data alone. Moreover, image flow alone cannot be recover the third component of range flow. Thus, when initialized, direct regularization with the combined regularized flow obtains the best result.

A 2002 journal article [3] by Spies et al. synthesized most of their previously published work from conference papers. They described 3D range flow fields that can be computed from a sequence of range data. In addition, they showed how a local velocity estimate could be obtained in a total least squares or least square framework. This technique distinguishes areas where line and plane normal as well as full range flow can be computed. They then showed how the sparse information from the TLS technique could be regularized to yield a dense full flow field. Moreover, they explain the direct regularization and the combination between intensity and depth data to gain 3D range flow. The performance of these algorithms was quantitatively assessed on synthetic and real data and the method was found to give excellent results. Finally, they showed that the motion of living castor bean leaves can be nicely captured.

2.2 Related Work

Scene flow and range flow both compute 3D motion of the object's surface; however, they use different concepts. We observe that most of the research in this field alternates between these two terms. Hence, in the range flow section, we note if the studies used range flow constraint equation in their method, when they used the depth as a required input of the functional, and if they applied derivatives on the depth directly. Conversely, in scene flow section, we observe that other 3D motion estimation studies use various methods, including first computing 2D optical flow and then using the disparity to estimate the third component.

2.2.1 Range Flow Studies

Yamamoto et al., in 1993, described a method that could estimate the range flow from a range sequence of non-rigid and rigid objects [100]. By solving a scheme of linear equations acquired in the substitution of a linear conversion of non-rigid subjects expressed by the Jacobian matrix, this technique may directly estimate deformable movement parameters. The algorithm has been tested using real sequences of paper, cloth, human skin, and a rubber balloon acquired from videos of range cameras.

In 1999, Harville et al. [101], explored the direct motion estimation with depth videos using a combination of range and intensity constraint equations. They tracked the pose of faces in sequences of synthetic and real images.

Imiya and Yamada, in 2006, [102] developed statistical methods to compute 3D range flow by formalizing the linear flow field as a model-fitting that can be solved by least square. They use random-sampling-and-voting method for the computation of optical flow as a model-fitting problem.

In 2007 [103], Matzaka et al. estimated translation 3-D motion from range images sequences using Predictive Motion Vector Field Adaptive Search Technique (PMVFAST). It was evaluated on both synthetic and real-live range image data acquired by a PMD (Photonic Mixing Device).

Schmidt et al. [104] estimated 3D range flow field from range data acquired using a Time-of-Flight camera. They derived a new variant of the range flow constraint equation that directly incorporated the transformation from the sensor to the world coordinate system using a pinhole camera model. To solve it, they applied the TLS approach. This method tested intensity and range data acquired from PMD.

In 2008, Spies and Barron's range flow estimation method [3] was extended by Schuchert et al.[105] to handle brightness changes in image data caused by inhomogeneous illumination through pre-filtering the intensity data using highpass or a homomorphic filter. In [106], they applied the algorithm using synthetic and real scenarios. In [107], they extended his work in [108, 109] and demonstrated how a standard affine optical flow model can be extended to 4D for estimation of 3D object structure, 3D motion, and rotation using a 1D camera grid. This camera grid equidistantly samples a 4D space spanned by the sensor coordinates x and y, camera position s, and time t. With this 4D data, optical flow and 3d range flow can be calculated

using TLS. This model has been tested using real data of a plant physiology experiment.

Another study in 2010 by Yu and Lang [110] estimated 3D local range flow using surface isometry as a primary constraint. Range flow constraint equation was used only for a final verification step. They applied GPU LK (Lucas/Kanade). They successfully found a dense range flow in two sequences in which paper was bent over 14 frames and the deformation of a plush toy dinosaur over eight frames.

In 2011, Gottfried et al. [111] introduced a method that computes 3D range flow using Kinect V1. They begin by calibrating the Kinect V1 and then aligning colour and depth images. They used direct regularization approaches as proposed by Spies et al. [3]. This study provided a general and robust solution to register the colour and depth channels. Their method for range flow estimation also provided stable flow fields.

In 2012 [112], Francis et al. proposed a method of estimating 3D range flow using depth information acquired using a single PMD camera. They utilized the combined Lucas/Kanade and Horn/Schunck techniques, which showed encouraging results for the use of this combined method.

In the same year, Ghuffar et al. [113] explored the 3D motion and segmentation of independently moving objects in range videos using a Time-of-Flight (TOF) camera. They used the algorithm based on the fusion of range flow and optical flow constraint equations [99] for both TLS and direct regularization [89]. They then used 3D motion to determine the trajectories and group them into spatio-temporal segments. It was tested in a real-world scene captured by a Time-of-Flight camera.

Also in 2012, Lukins and Fisher [114] added colour channels (not only intensity) to the depth information to estimate 3D range flow using the least-square method. They claimed that this step could improve the estimation and resulting flow in synthetic and real data.

Birdal [115], in the final notable study from 2012, added colours (not only intensity) to the range flow estimation. They estimated 3D range flow by utilizing the combined LK and HS approach. They also introduced a framework that automatically initializes the interest area by segmenting of the objects using Kinect images. This proposed method was also tested using synthetic and real data.

In 2013, Jones [116, 117, 118] combined optical flow and 3D range flow in a real-time framework to estimate the translation and rotation of ego-motion of the RGB-D sensor using least square. The algorithm was evaluated on the TUB RGB-D Benchmark and used Kinect V1 to generate this dataset.

Ghuffar et al. [119], in 2013, estimated the ego-motion of the TOF camera using the fusion of optical flow and range flow constraints in the LS approach. The videos involve translation and rotation motions. Then, [120] used this method to segment independently moving objects out of the camera motion in real scenarios.

Also in 2013, Okorn and Harguess [121] applied the high-order polynomial extension to estimate 3D range flow using the non-linear LS method. It was tested using data from a VelodyneR HDL-64E sensor. Then, in [122], they estimated the local and global 3D range flow of the ego-

motion of the sensor in a sequence of range data.

Herbst et al. [123], again in 2013, combined color and depth images with a smoothness term in a variational model to estimate 3D scene flow. This method segmented rigid objects and was tested using real data.

Quiroga et al. [124] proposed a method to compute 3D range flow by tracking surface patches using the Lucas/Kanade framework in an aligned pair of intensity and depth images from the Kinect. They tested this method using synthetic data and a sequence from Kinect V1. In a subsequent study, Quiroga et al. [125] presented another method by combining local and global constraints in a variational framework. In this work, depth data is used in 3 ways: to model the motion in the image domain, to constrain the scene flow, and to adapt the TV regularization. In addition, they constrained the motion using a set of 3D correspondences to account for large displacements. They tested this method using Kinect V1 images. In 2014, Quiroga et al. [126, 127] improved his algorithm by using TV (Total Variation) in the piecewise smooth regularization. They decoupled the regularization procedure for the rotation and translation motion. An iteratively reweighted LS method was used to estimate local rigid motion. Experiments confirm the ability of these algorithims to correctly estimate the flow in synthetic and real-world scenes.

Jaimez et al. [128], in 2015, proposed a primal-dual algorithm implemented in GPU to estimate 3D scene flow for RGB-D cameras in real-time. Their variational framework imposed brightness constancy and geometric consistency. They applied Total Variation (TV) regularization and provided quantitative and qualitative results from this semi-real data. They generated artificial RGB-D images from a previously captured RGB-D frame and then defined 3D motion field. Results show that the output motion is capable of accurately estimating non-rigid motions at a high frame rate. Finaly, [129] introduced a MC (motion cooperation) method to estimate 3D scene flow by combining registration and segmentation approaches.

In 2018 [130], Xiang et al. estimated 3D flow using aligned colour and depth map images in a variational framework. They applied the 3D local rigidity assumption in the data term. To preserve the motion in the boundaries, they used the depth map as a weighted anisotropic diffusion tensor in the smoothness term. They tested this method using Middlebury data sets and real Kinect data.

2.2.2 Scene Flow Studies

Scene flow term was first introduced by Vedula et al. in 1999 [131]. They computed non-rigid 3D scene flow using 2D optical flow by applying the linear algorithm. They used a hierarchical version of the Lucas/Kanade algorithm to estimate 2D optical flow. In 2005, the same authors [132] improved their algorithms by using regularization in the image plane. They introduced three algorithms using a single with known geometry or multiple cameras with known and unknown geometry.

Ye Zhang and Chandra Kambhamettu [133, 134, 135], between 2001 to 2003, presented two algorithms that compute dense 3D scene flow from multi-view image sequences. In the first approach, they assumed that each small region has similar motion, then applied a non-linear

model to each region to estimate 3D motion and structure. In the second approach, they used a hierarchical approach in a stereo matching algorithm. Experimental results on both synthetic and real imagery showed the effectiveness of the algorithms.

In 2003, Pons et al. [136] presented a variational framework for dense depth recovery and dense 3D motion field estimation from multiple video sequences. The authors improved this algorithm in 2007 [137] as a method for multi-view stereovision and non-rigid 3D dimensional motion estimation from multiple video sequences. This algorithm yields good results on a variety of datasets, including specularities and translucency. They have also successfully tested this motion estimation algorithm on some challenging multi-view video sequences of a non-rigid scene.

Isard and MacCormick [138], in 2006, presented an estimation of a scene flow algorithm. They computed a dense estimate of motion and disparity given a stereo video sequence containing moving non-rigid objects. This method estimated motion using loopy belief propagation. The results demonstrated that the simultaneous estimation of motion and disparity is superior to using stereo alone.

Another scene flow algorithm published in 2006 by Devernay et al.[139] computed scene flow by tracking 3-D points and surface elements in a multi-camera setup using the Lucas-Kanade tracking algorithm. This method gives real-time execution.

In 2007, Huguet and Devernay [140] presented a variational framework to compute scene flow from a stereoscopic image sequence. They coupled 2D optical flow in both cameras with dense stereo matching between the images. This method used the same framework presented by Brox et al. 2004 [6]; however, they added epipolar geometry constraints, and showed that the same kind of numerical solution could be used. The experiments showed that the method can handle real stereo sequences with large motion and stereo discontinuities. They also tested their algorithm in synthetic data with ground truth.

Li and Sclaroff [141], also in 2007, proposed an algorithm that estimates 3D scene flow using only two cameras by combining stereo and optical flow into a single framework. In their method, they resolve the uncertainty from 2D computation by using region information to regularize the 2D estimates.

In 2008, Wedel et al. [142, 143, 144] proposed a variational framework that can estimate 3D scene flow and depth information using a stereo image sequences. They partially separated the depth estimation from the motion estimation step. Their approach was implemented in a GPU that allows for real-time computation. In 2009, Wedel et al. [145] presented an approach for identifying and segmenting independently moving objects from dense scene flow information by using a moving stereo camera system. Their results systematically demonstrated the improvement in reliability measures for the scene flow variables.

Rabe et al. [146] extended Wedel et al. work's in 2010 using Kalman filtering for temporal smoothness and robustness. They implemented this on a GPU yielding real-time computation. [147] presented a variational method to estimate 3D scene flow in uncalibrated stereo sequences by assuming that only the intrinsic camera parameters were known. They integrated the spatial and temporal information from two stereo pairs in global energy functional.

In 2011, Cech et al.[148] proposed a simple seed growing algorithm for estimating scene flow in stereo images. They started with corresponding seeds to estimate the disparity map between image pair and calculate 2D optical flow for consecutive images. They tested the algorithm with two different datasets and performed a quantitative ground truth experiment.

Vogel et al. [149] presented an approach for 3D scene flow estimation. They showed that standard smoothness priors from the 2D motion estimation lead to biases when applied to 3D scene flow estimation. To address this issue, their method regularizes 3D scene flow computation by penalizing deviations from the local rigidity of the motion and integrating it into an energy minimization framework. Their experiments on several datasets demonstrates reductions of the 3D motion estimation error compared to standard total variation regularization and shows the applicability to real-world scenes with articulated motion.

Also in 2011, Hadfield and Bowden [150] demonstrated a novel formulation for scene flow estimation using a collection of moving points in 3D space modelled with a particle filter that supports multiple hypotheses and does not cover smooth the motion field. Unlike traditional approaches, the algorithm is capable of operating on single viewpoint sequences. It is one of the few scene flow estimation systems capable of using modern depth sensor technology such as the Kinect, rather than relying on stereo matching algorithms. The algorithm was applied to an existing scene flow data set, where it achieved comparable results to approaches that use multiple views.

Letouzey et al. [151] estimated 3D motion using photometric constraints and a global regularization term. This approached required a depth camera and one or more color cameras. They were tested in synthetic and real data.

Sizintsev et al. [152] introduced spatiotemporal quadric element (stequels) matching to estimate 3D scene flow. Their algorithm was tested using two stereo sequences taken by a BumbleBee stereo camera.

In 2013, Basha et al. [153] published a novel approach in scene flow estimation. They proposed a new variational approach for simultaneously estimating the scene flow and structure from calibrated multi-view sequences. Their proposed global energy functional incorporates the information from available sequences and simultaneously recovers both depth and scene flow. The functional enforces multi-view geometric consistency and imposes brightness constancy and piecewise smoothness assumptions directly on the 3D unknowns. The new 3D unknowns rely on a 3D point cloud representation and smoothness assumptions. It inherently handles the challenges of discontinuities, occlusions, and large displacements. This variational framework, used for the first time for multiple views and 3D representation, successfully recovers the 3D structure and scene flow. This method shows accurate and dense results on real and synthetic data.

Vogel et al. [154, 155] introduced a model that represents the 3D scene flow through a collection of planar, rigidly moving, local segments. The energy function combines an occlusion data term with appropriate shape, motion, and segmentation regularizers. They tested this method using different datasets, including KITTI benchmark.

Hadfield and Bowden, in 2014 [156], estimated the 3D scene flow using particle filtering in a

3D coloured point cloud representation of the scene. The algorithm applied several hypotheses to resolve motion ambiguities for each 3D point. The approach was applied to 3D hand tracking during sign language.

In 2015, Sun et al. [157] proposed a layered RGB-D scene flow method that jointly solves for the scene segmentation and the motion in Kinect V1 imaging data. They used depth images to decide the depth ordering of layers and estimate the 3D scene flow for each layer. Experiments were done in both Middlebury and real-world sequences.

Zanfi et al. [158], also in 2015, proposed a 3D scene flow estimation method from a single view that can handle large displacement using the coarse-to-fine framework. In this method, they integrate geometric and photometric components into one constant. They also applied an occlusion aware energy model, defined over-lapping, and image-adaptive neighbourhoods to process fast motions and large occlusions areas. It was tested using different CAD 120 datasets and MPI Sintel dataset.

In 2016, Richard et al. [159] estimated 3D dense scene flow from two handheld videos captured with wide camera baseline and different camera and sensor characteristics. This method can estimate the motion for independently moving cameras.

Also, in 2016, Lv et al. [160] introduced a continuous optimization method that can estimate 3D scene flow from stereo images. They solve the pixel-to-plane assignment and plane-to-motion assignment in a continous formulation. This method was evaluated using KITTI Scene Flow benchmark.

KITTI datasets were first published by Geiger et al. in 2012 [32]; they subsequently published another challenging set in 2015. They recorded an outdoor environment using four high resolution video cameras, a Velodyne laser scanner, and localization system. These datasets are available for different tasks, such as extracting disparity from the stereo, optical flow, visual odometry/SLAM, and 3D object detection. Recently, several studies for 3D scene flow have used these datasets to evaluate their methods.

In 2017, Taniai et al. [161] presented a method that can estimate 3D scene flow and segment rigid objects using multi-frames. They estimated the disparity map and the 6-DOF camera motion using stereo matching and visual odometry. They subsequently computed optical flow in the regions that are inconsistent with camera motion. This method showed good results in KITTI data sets.

Xiao et al. [162] estimated scene flow using a monocular camera by combining an occlusion function with scene flow energy functional. An inverse depth technique was used to put the depth in the functional.

Another study in 2018 by Ren et al.[163] studied scenes consisting of a fixed background with rigidly objects moving in the front; consequently, they used semantic cues to produce scene flow estimation. This cascaded classification framework models 3D scene flow by iteratively refining semantic segmentation masks, stereo correspondences, 3D rigid motion estimates, and optical flow fields. This method was tested using KITTI autonomous driving benchmark.

Liu et al., in 2018, [164] estimated 3D scene flow directly from point clouds. They proposed

a deep neural network named FlowNet3D that learned scene flow from point clouds. This method was tested on synthetic data and KITTI data sets.

Also in 2018, Menze et al.[87, 165] published a study that estimated 3D scene flow in addition to the location, shape, and motion of vehicles in stereo images. It assumed that the outdoor scene consists of moving rigid objects. They use a slanted-plane model; in other words, they assume that the 3D structure of the scene can be approximated by a set of piecewise planar superpixels. They tested their method using KITTI datasets.

Xiang et al. [166] proposed a method to estimate 3D dense scene flow in stereo image sequences using a bilateral filter and an adaptive TV (Total Variation) penalty function. They tested this method using real stereo data set provided by Middlebury and a synthetic driving sequence.

Lastly, Schuster et al., in 2017, [167] published a study that estimated 3D scene flow according to dense interpolation of sparse matches across two stereo images. Their method combines multi-scale matching and edge-preserving interpolation. A few iterations of variational energy minimization refined the results. They named their method SceneFlowFields. In [168], they combined the 2D optical flow (Flow Field+) method [169] with stereo disparity Semi-Global Matching (SGM) [170] algorithms to achieve 3D scene flow. In 2019, Schuster et al. [171] improved SceneFlowFields into SceneFlowFields++ by adding multi-frame matching with visibility prediction and robust estimation of a 3D geometry for a small superpixels motion. All their studies were tested using KITTI datasets and MPI Sintel datasets.

2.3 Conclusion

This chapter defined 2D/3D optical flow and described their motion constraint equations and the aperture problem. Several 2D/3D approaches were briefly mentioned. Scene flow and range flow can be considered special cases in 3D optical flow. Motion constraint equations and 3D aperture problem were discussed in terms of range flow cases. The last part described Spies and Barron's range flow algorithms in more detail because they are central to our research. Lastly, the chapter provided a brief related work section about range flow and scene flow studies. The next chapter explains our implementation of 2D and 3D approaches in extensive mathematical detail.

Chapter 3

Theoretical Technique

This chapter discusses the approaches that we implemented to estimate 2D optical flow and 3D range flow in mathematical detail. We implemented 18 approaches (with and without robust technique) to determine the flow. We started with local approaches using least square and total least square and global approaches (regularization), before moving into combined Local and Global (CLG) approaches as per Brox et al. These algorithms describe both 2D optical flow using intensity data and also 3D range flow using depth data. In addition, we show how to calculate 3D range flow when combining intensity and range data together. We implement two versions with and without robust technique. This chapter explains the mathematical details of the versions without robust technique; importantly, robust has similar derivatives when adding robust estimator term. Therefore, we provide the final matrix formula with robust estimator for each approach. Moreover, we show the output flow for NSERC dataset for each algorithm.

3.1 Local Approaches:

3.1.1 Optical Flow by Least Square (LS_i)

This is Lucas and Kanade's technique [1]. Using the motion constraint equation:

$$I_x u + I_y v + I_t = 0 \tag{3.1}$$

which is one equation in two unknowns, $\vec{v} = (u, v)$. In each local $n \times n$ neighbourhood, we obtain up to n^2 equations, if we assume the neighbourhood moves with constant velocity. We obtain a $n \times 2$ linear system of equations:

$$A_{n\times 2}\vec{v} = B_{n\times 1}, \tag{3.2}$$

where A:

$$A = \begin{bmatrix} < I_x I_x > & < I_x I_y > \\ < I_x I_y > & < I_y I_y > \end{bmatrix} \tag{3.3}$$

and B:

$$B = \begin{bmatrix} < I_x I_t > \\ < I_y I_t > \end{bmatrix} \tag{3.4}$$

$\vec{v}$ can be computed via the least squares as:

$$\vec{v} = (A^T A)^{-1} A^T B. \tag{3.5}$$

The eigenvectors $\hat{e}_0$ and $\hat{e}_1$ and their corresponding eigenvalues λ_0 and λ_1 can be computed from the 2×2 symmetric matrix $(A^T A)$ and used to compute the least squares full image velocity when $\lambda_0, \lambda_1 > \tau$ or the least squares normal velocity when $\lambda_1 > \tau, \lambda_0 \leq \tau$. τ is a threshold value need to specify.

Raw plane normal velocity can also be directly computed from intensity derivatives as

$$\vec{v_n} = \frac{-I_t(I_x, I_y)}{\sqrt{I_x^2 + I_y^2}}. \tag{3.6}$$

For robust estimation, we are using ψ function for data term.

$$\psi_{data}\left((I_x u + I_y v + I_t)^2\right) \tag{3.7}$$

In this case A and B become:

$$A = \begin{bmatrix} \psi'_{data} * < I_x I_x > & \psi'_{data} * < I_x I_y > \\ \psi'_{data} * < I_x I_y > & \psi'_{data} * < I_y I_y > \end{bmatrix} \tag{3.8}$$

and B:

$$B = \begin{bmatrix} \psi'_{data} * < I_x I_t > \\ \psi'_{data} * < I_y I_t > \end{bmatrix} \tag{3.9}$$

Figure 3.1 shows the computed 2D optical flow in NSERC datasets using least square with intensity data only.

3.1.2 Range Flow by Least Square (LS_r)

Similar to the previous method, Spies and Barron [3] used range motion constraint equation to obtain 3D range flow:

$$Z_x u + Z_y v + Z_z w + Z_t = 0. \tag{3.10}$$

For range data $Z_z = 1$ and the range constraint equation becomes:

$$Z_x u + Z_y v + w + Z_t = 0. \tag{3.11}$$

As for optical flow, we can compute $A_{n\times 3}\vec{V} = B_{n\times 1}$, where A:

$$A = \begin{bmatrix} < Z_x Z_x > & < Z_x Z_y > & < Z_x > \\ < Z_x Z_y > & < Z_y Z_y > & < Z_y > \\ < Z_x > & < Z_y > & 1 \end{bmatrix} \tag{3.12}$$

and B:

$$B = \begin{bmatrix} < Z_x Z_t > \\ < Z_y Z_t > \\ < Z_t > \end{bmatrix} \tag{3.13}$$

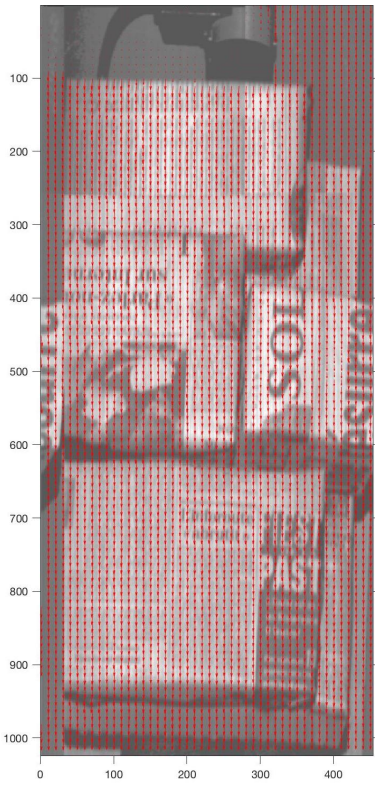


Figure 3.1: 2D optical flow using intensity data with LS method

The least squares solution for $\vec{V} = (u, v, w)$ is:

$$\vec{V} = (A^T A)^{-1} A^T B. \tag{3.14}$$

The eigenvectors $\hat{e}_0$, $\hat{e}_1$ and $\hat{e}_2$ and their corresponding eigenvalues, λ_0, λ_1 and λ_2 can be computed from the 3×3 symmetric matrix $A^T A$ and can be used to compute the least squares full range velocity, $\vec{V}_F$, when $\lambda_0, \lambda_1, \lambda_2 > \tau$, the least squares line normal velocity, $\vec{V}_L$, when $\lambda_1, \lambda_2 > \tau$, $\lambda_0 \leq \tau$ and least squares plane normal velocity, $\vec{V}_P$, when $\lambda_2 > \tau$, $\lambda_0, \lambda_1 \leq \tau$. τ is a threshold values need to specify to determine the flow. That is:

$$\vec{V}_F = (\vec{V} \cdot \hat{e}_0)\hat{e}_0 + (\vec{V} \cdot \hat{e}_1)\hat{e}_1 + (\vec{V} \cdot \hat{e}_2)\hat{e}_2, \tag{3.15}$$

$$\vec{V}_L = (\vec{V} \cdot \hat{e}_1)\hat{e}_1 + (\vec{V} \cdot \hat{e}_2)\hat{e}_2, \tag{3.16}$$

$$\vec{V}_P = (\vec{V} \cdot \hat{e}_2)\hat{e}_2. \tag{3.17}$$

Raw plane normal velocity can also be directly computed from Z derivatives as:

$$\vec{V}_R = \frac{-Z_t(Z_x, Z_y)}{\sqrt{Z_x^2 + Z_y^2 + 1}} \tag{3.18}$$

For the robust estimation, we are using ψ function for data term.

$$\psi_{data}\left((Z_x u + Z_y v + w + Z_t)^2\right) \tag{3.19}$$

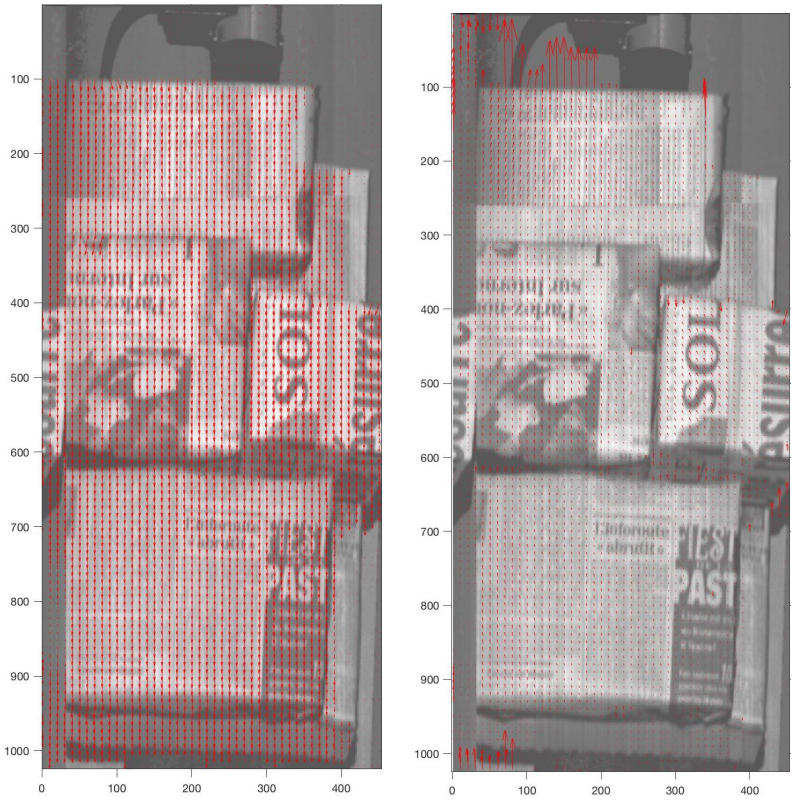
In this case, A and B become:

$$A = \begin{bmatrix} \psi'_{data} * < Z_x Z_x > & \psi'_{data} * < Z_x Z_y > & \psi'_{data} * < Z_x > \\ \psi'_{data} * < Z_x Z_y > & \psi'_{data} * < Z_y Z_y > & \psi'_{data} * < Z_y > \\ \psi'_{data} * < Z_x > & \psi'_{data} * < Z_y > & \psi'_{data} * 1 \end{bmatrix} \tag{3.20}$$

and B:

$$B = \begin{bmatrix} \psi'_{data} * < Z_x Z_t > \\ \psi'_{data} * < Z_y Z_t > \\ \psi'_{data} * < Z_t > \end{bmatrix} \tag{3.21}$$

Figure 3.2 shows the computed 3D range flow in NSERC datasets using the least square method with range data only.



(a) (b)

Figure 3.2: 3D range flow using range data with LS method (a) u,v components of 3D range flow, and (b) u,w components

3.1.3 Range Flow by Least Square Using Intensity and Range data (LS_ir)

It is possible to compute (u, v, w) using both intensity and range derivatives by combining the intensity constraint equation and range constraint equation. We need to solve:

$$\int\int\int \beta^2 (I_x u + I_y v + I_t) + (Z_x u + Z_y v + w + Z_t) dx dy dz dt \rightarrow 0. \tag{3.22}$$

This yields the linear equation:

$$(\beta^2 I_x + Z_x)u + (\beta^2 I_y + Z_y)v + w + (\beta^2 I_t + Z_t) = 0, \tag{3.23}$$

which can be solved in local $(n \times n)$ neighbourhoods with a least squares calculation. β^2 is a weighting term that can take into account the relative magnitude of the intensity and range derivatives.

We can compute $A_{n\times 3}\vec{V} = B_{n\times 1}$, where A:

$$A = \begin{bmatrix} < Z_x Z_x + \beta^2 I_x I_x > & < Z_x Z_y + \beta^2 I_x I_y > & < Z_x > \\ < Z_x Z_y + \beta^2 I_x I_y > & < Z_y Z_y + \beta^2 I_y I_y > & < Z_y > \\ < Z_x > & < Z_y > & 1 \end{bmatrix} \tag{3.24}$$

and B:

$$B = \begin{bmatrix} < Z_x Z_t + \beta^2 I_x I_t > \\ < Z_y Z_t + \beta^2 I_y I_t > \\ < Z_t > \end{bmatrix} \tag{3.25}$$

The least squares solution for $\vec{V} = (u, v, w)$ is defined using equations (3.14), (3.15), (3.16) and (3.17).

For robust estimation, we are using ψ function for data term.

$$\psi_{data}\left((\beta^2 I_x + Z_x)u + (\beta^2 I_y + Z_y)v + w + (\beta^2 I_t + Z_t)^2\right) \tag{3.26}$$

In this case A and B become:

$$A = \begin{bmatrix} \psi'_{data} * < Z_x Z_x + \beta^2 I_x I_x > & \psi'_{data} * < Z_x Z_y + \beta^2 I_x I_y > & \psi'_{data} * < Z_x > \\ \psi'_{data} * < Z_x Z_y + \beta^2 I_x I_y > & \psi'_{data} * < Z_y Z_y + \beta^2 I_y I_y > & \psi'_{data} * < Z_y > \\ \psi'_{data} * < Z_x > & \psi'_{data} * < Z_y > & \psi'_{data} * 1 \end{bmatrix} \tag{3.27}$$

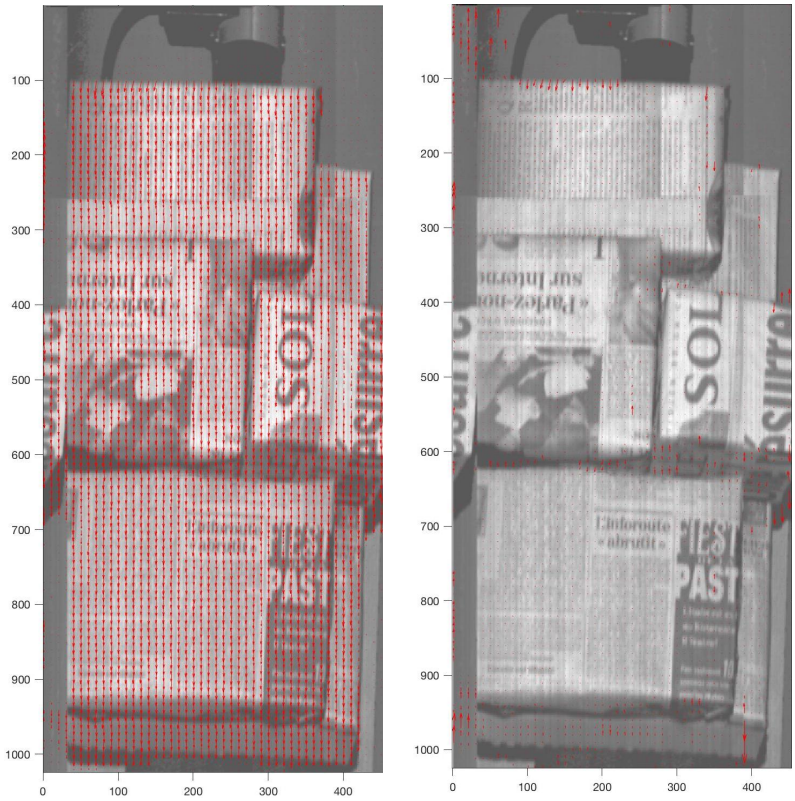
and B:

$$B = \begin{bmatrix} \psi'_{data} * < Z_x Z_t + \beta^2 I_x I_t > \\ \psi'_{data} * < Z_y Z_t + \beta^2 I_y I_t > \\ \psi'_{data} * < Z_t > \end{bmatrix} \tag{3.28}$$

Figure 3.3 shows the computed 3D range flow in NSERC datasets using the least square method by combining intensity and range data together.

3.1.4 Optical Flow by Total Least Square (TLS_i)

Instead of computing optical flow using least square only, we can add the right-hand side B in our consideration. Spies et al. [172] computed 2D optical flow using structure tensor as proposed by Jahen et al. in 1998 [173]. They are computing the eigenvalues and eigenvectors of the 3×3 structure tensor J of each pixel to estimate 2D optical flow.



(a) (b)

Figure 3.3: 3D range data using intensity/range data with the LS method (a) u,v components of 3D range flow, and (b) u,w components

$$J = \begin{bmatrix} < I_x I_x > & < I_x I_y > & < I_x I_t > \\ < I_x I_y > & < I_y I_y > & < I_y I_t > \\ < I_x I_t > & < I_y I_t > & < I_t I_t > \end{bmatrix} \tag{3.29}$$

The eigenvectors $\hat{e}_0$, $\hat{e}_1$ and $\hat{e}_2$ and their corresponding eigenvalues, λ_0, λ_1 and λ_2 can be computed from the 3×3 matrix J and can be used to compute the total least squares full image velocity, when $\lambda_0 < \tau, \lambda_1, \lambda_2 > \tau$, the total least squares normal velocity when $\lambda_1 < \tau, \lambda_2 > \tau$. The τ is a threshold values need to specify to determine the flow.

Optical flow $\vec{v} = (u, v)$ can be computed using the smallest eigenvector $\hat{e}_3$ corresponding to the smallest eigenvalue λ_3 of the matrix as follows:

$$\vec{f_f} = \frac{1}{e_{13}} \begin{bmatrix} e_{11} \\ e_{12} \end{bmatrix} \tag{3.30}$$

For normal flow :

$$\vec{f_n} = \frac{e_{33}}{e_{31}^2 + e_{32}^2} \begin{bmatrix} e_{31} \\ e_{32} \end{bmatrix} \tag{3.31}$$

For robust estimation, we are using ψ function for data term.

$$\psi_{data}\left((I_x u + I_y v + I_t)^2\right) \tag{3.32}$$

In this case J becomes:

$$J = \begin{bmatrix} \psi'_{data} * < I_x I_x > & \psi'_{data} * < I_x I_y > & \psi'_{data} * < I_x I_t > \\ \psi'_{data} * < I_x I_y > & \psi'_{data} * < I_y I_y > & \psi'_{data} * < I_y I_t > \\ \psi'_{data} * < I_x I_t > & \psi'_{data} * < I_y I_t > & \psi'_{data} * < I_t I_t > \end{bmatrix} \tag{3.33}$$

Figure 3.4 shows the computed 2D optical flow in NSERC datasets using the total least square method with intensity data only.

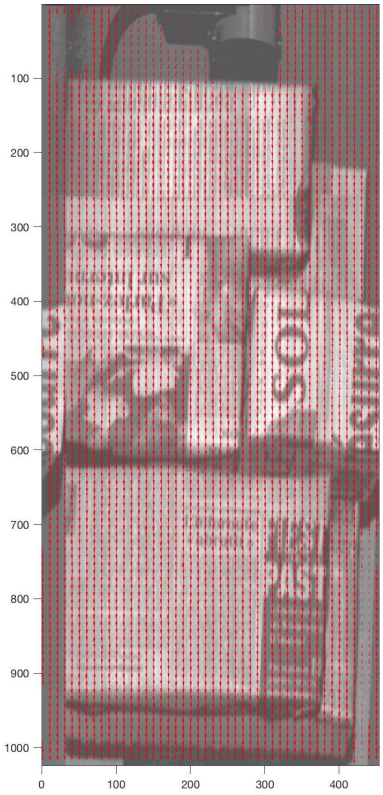


Figure 3.4: 2D Optical flow using intensity data with TLS method

3.1.5 Range Flow by Total Least Square (TLS_r)

Spies and Barron, in [3], extend the total least square method to estimate 3D range flow similar to the previous method. The solution of $\vec{V} = (u, v, w, 1)$ is given by the eigenvector $\hat{e}_4$, corresponding to the smallest eigenvalue λ_4 of the generalized 4×4 structure tensor J:

$$J = \begin{bmatrix} < Z_x Z_x > & < Z_x Z_y > & < Z_x > & < Z_x Z_t > \\ < Z_y Z_x > & < Z_y Z_y > & < Z_y > & < Z_y Z_t > \\ < Z_x > & < Z_y > & < 1 > & < Z_t > \\ < Z_t Z_x > & < Z_t Z_y > & < Z_t > & < Z_t Z_t > \end{bmatrix} \tag{3.34}$$

The eigenvectors $\hat{e}_0$, $\hat{e}_1$, $\hat{e}_2$ and $\hat{e}_3$ and their corresponding eigenvalues, λ_0, λ_1, λ_2 and λ_3 can be computed from the 4×4 matrix J. The flow can be estimated when the trace is $> \tau 1$. Thus, the eigenvectors and eigenvalues can be used to compute the total least squares full range velocity, $\vec{V}_F$, when $\lambda_0 < \tau 2, \lambda_1 > \tau 3$, line normal velocity, $\vec{V}_L$, when $\lambda_1 < \tau 3, \lambda_2 > \tau 3$ and total least squares plane normal velocity, $\vec{V}_P$, when $\lambda_2 < \tau 3$, $\lambda_3 > \tau 3$. The $\tau 1, \tau 2, \tau 3$ are threshold values need to specify to determine the flow.

The various range flow types (full, line, plane) can be detected from the eigenvalues of matrix J as follows:

The sought full flow is found to be:

$$\vec{f_f} = \frac{1}{e_{14}} \begin{bmatrix} e_{11} \\ e_{12} \\ e_{13} \end{bmatrix} \tag{3.35}$$

The plane flow is:

$$\vec{f_p} = \frac{e_{44}}{e_{41}^2 + e_{42}^2 + e_{43}^2} \begin{bmatrix} e_{41} \\ e_{42} \\ e_{43} \end{bmatrix} \tag{3.36}$$

and the line flow is:

$$\vec{f_l} = \frac{e_{41}}{1 - e_{34}^2 - e_{44}^2} \left[e_{34} \begin{bmatrix} e_{31} \\ e_{32} \\ e_{33} \end{bmatrix} + e_{44} \begin{bmatrix} e_{41} \\ e_{42} \\ e_{43} \end{bmatrix} \right] \tag{3.37}$$

This method can distinguish areas where line and plane normal, as well as full range flow, can be computed.

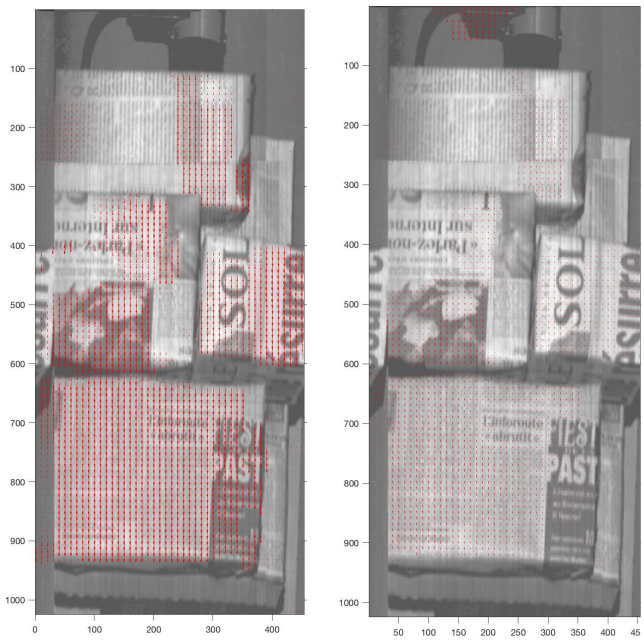
For robust estimation, we are using ψ function for data term.

$$\psi_{data} \left((Z_x u + Z_y v + w + Z_t)^2 \right) \tag{3.38}$$

In this case J becomes:

$$J = \begin{bmatrix} \psi'_{data} * < Z_x Z_x > & \psi'_{data} * < Z_x Z_y > & \psi'_{data} * < Z_x > & \psi'_{data} * < Z_x Z_t > \\ \psi'_{data} * < Z_y Z_x > & \psi'_{data} * < Z_y Z_y > & \psi'_{data} * < Z_y > & \psi'_{data} * < Z_y Z_t > \\ \psi'_{data} * < Z_x > & \psi'_{data} * < Z_y > & \psi'_{data} * < 1 > & \psi'_{data} * < Z_t > \\ \psi'_{data} * < Z_t Z_x > & \psi'_{data} * < Z_t Z_y > & \psi'_{data} * < Z_t > & \psi'_{data} * < Z_t Z_t > \end{bmatrix} \tag{3.39}$$

Figure 3.5 shows the computed 3D range flow in NSERC datasets using the total least square method with range data only.



(a) (b)

Figure 3.5: 3D range flow using range data with TLS method (a) u,v components of 3D range flow, and (b) u,w components

3.1.6 Range Flow by Total Least Square Using Intensity and Range data (TLS_ir)

It is possible to compute range flow $\vec{V} = (u, v, w, 1)$ using both intensity and range derivatives. Generalize 4×4 structure tensor would be:

$$J = \begin{bmatrix} < Z_x^2 + I_x^2 > & < Z_x Z_y + I_x I_y > & < Z_x > & < Z_x Z_t + I_x I_t > \\ < Z_x Z_y + I_x I_y > & < Z_y^2 + I_y^2 > & < Z_y > & < Z_y Z_t + I_y I_t > \\ < Z_x > & < Z_y > & < 1 > & < Z_t > \\ < Z_t Z_x + I_t I_x > & < Z_t Z_y + I_t I_y > & < Z_t > & < Z_t^2 > \end{bmatrix} \tag{3.40}$$

To compute full, line, and plane flow, we can use the same respective formulas (3.35), (3.36) and (3.37).

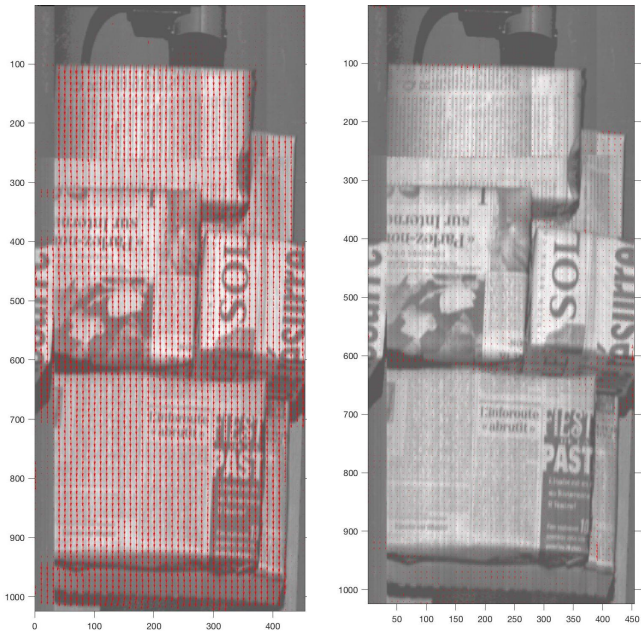
For robust estimation, we are using ψ function for data term.

$$\psi_{data}\left((\beta^2 I_x + Z_x)u + (\beta^2 I_y + Z_y)v + w + (\beta^2 I_t + Z_t)^2\right) \tag{3.41}$$

In this case J become:

$$J = \begin{bmatrix} \psi'_{data} * < Z_x^2 + I_x^2 > & \psi'_{data} * < Z_x Z_y + I_x I_y > & \psi'_{data} * < Z_x > & \psi'_{data} * < Z_x Z_t + I_x I_t > \\ \psi'_{data} * < Z_x Z_y + I_x I_y > & \psi'_{data} * < Z_y^2 + I_y^2 > & \psi'_{data} * < Z_y > & \psi'_{data} * < Z_y Z_t + I_y I_t > \\ \psi'_{data} * < Z_x > & \psi'_{data} * < Z_y > & \psi'_{data} * < 1 > & \psi'_{data} * < Z_t > \\ \psi'_{data} * < Z_t Z_x + I_t I_x > & \psi'_{data} * < Z_t Z_y + I_t I_y > & \psi'_{data} * < Z_t > & \psi'_{data} * < Z_t^2 > \end{bmatrix} \tag{3.42}$$

Figure 3.6 shows the computed 3D range flow in NSERC datasets using the total least square method with intensity and range data together.



(a) (b)

Figure 3.6: 3D range flow using intensity/range data with TLS method(a) u,v components of 3D range flow, and (b) u,w components

3.2 Global Approaches(Regularization):

In regularization approaches, we need to minimize the energy functional using Euler-Lagrange equation. It is the major formula in the calculus of variations. The general formulas to estimate (u,v) are:

$$f_u - \frac{df_{u_x}}{dx} - \frac{df_{u_y}}{dy} = 0 \tag{3.43}$$

$$f_v - \frac{df_{v_x}}{dx} - \frac{df_{v_y}}{dy} = 0, \tag{3.44}$$

for 3D, the general formulas to compute (u,v,w) are:

$$f_u - \frac{df_{u_x}}{dx} - \frac{df_{u_y}}{dy} - \frac{df_{u_z}}{dz} - \frac{df_{u_t}}{dt} = 0, \tag{3.45}$$

$$f_v - \frac{df_{v_x}}{dx} - \frac{df_{v_y}}{dy} - \frac{df_{v_z}}{dz} - \frac{df_{v_t}}{dt} = 0, \tag{3.46}$$

$$f_w - \frac{df_{w_x}}{dx} - \frac{df_{w_y}}{dy} - \frac{df_{w_z}}{dz} - \frac{df_{w_t}}{dt} = 0. \tag{3.47}$$

These formulas end up with a PDE (Partial Differential Equation) system that need an iterative solution to converge. Jacobian, Gauss-Sedial, and SOR (Sucessive Over Relaxation) are examples of iterative solutions for linear systems. Our implementation uses the SOR method. We suppose our linear system as:

$$Ax = b,$$

In SOR, A matrix split in three parts.

$$A = U + L + D,$$

$$Ux + Lx + Dx = b$$

U is the upper triangular of the matrix, L is the lower triangular, and D is the diagonal.

$$(L + D)x = b - Ux$$

Using the last iteration solution to estimate the current iteration.

The general formula for SOR iterative solution is [174]:

$$x^{k+1} = \omega(D - \omega L)^{-1}b - (D - \omega L)^{-1}[\omega U + (\omega - 1)D]x^k$$

$\omega \in (0, 2)$ is a relaxation factor.

For each variable, we define the sequential solution as:

$$x_i^{k+1} = (1 - \omega)x^k + \frac{\omega}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}^{k+1} - \sum_{j=i+1}^{n} a_{ij}^{k}\right]$$

Regularization method has two different approaches [3]. Direct regularization is superior when we do not have initial values from any previous frameworks. The optical/range flow computed directly using the derivatives. Indirect regularization uses least square results as initial values.

3.2.1 Optical Flow by Regularization (Global_i)

This is Horn and Schunck's optical flow approach [2] using a motion constraint equation:

$$I_x u + I_y v + I_t = 0, \tag{3.48}$$

We want to regularize:

$$f(x, y, u, v, u_x, u_y, v_x, v_y) = (I_x u + I_y v + I_t)^2 + \alpha^2(u_x^2 + u_y^2 + v_x^2 + v_y^2) \tag{3.49}$$

by minimizing:

$$\int\int f(x, y, u, v, u_x, u_y, v_x, v_y)\partial x \partial y \tag{3.50}$$

We need to utilize Euler-Lagrange equations where:

$$f_u = 2I_x(I_x u + I_y v + I_t) \tag{3.51}$$

$$
\begin{aligned}
f_v &= 2I_y(I_x u + I_y v + I_t) && (3.52)\\
f_{u_x} &= 2\alpha^2 u_x && (3.53)\\
f_{u_y} &= 2\alpha^2 u_y && (3.54)\\
f_{v_x} &= 2\alpha^2 v_x && (3.55)\\
f_{v_y} &= 2\alpha^2 v_y && (3.56)\\
\frac{df_{u_x}}{dx} &= 2\alpha^2 u_{xx} && (3.57)\\
\frac{df_{u_y}}{dy} &= 2\alpha^2 u_{yy} && (3.58)\\
\frac{df_{v_x}}{dx} &= 2\alpha^2 v_{xx} && (3.59)\\
\frac{df_{v_y}}{dy} &= 2\alpha^2 v_{yy}. && (3.60)
\end{aligned}
$$

Since $\nabla^2 u = u_{xx} + u_{yy}$ and $\nabla^2 v = v_{xx} + v_{yy}$ we can rewrite the Euler-Lagrange equations as:

$$
\begin{aligned}
I_x^2 u + I_x I_y v + I_x I_t &= \alpha^2 \nabla^2 u && (3.61)\\
I_x I_y u + I_y^2 v + I_y I_t &= \alpha^2 \nabla^2 v. && (3.62)
\end{aligned}
$$

Using $\nabla^2 u \approx \bar{u} - u$ and $\nabla^2 v \approx \bar{v} - v$ we get

$$
\begin{aligned}
(\alpha^2 + I_x^2)u + I_x I_y v &= (\alpha^2 \bar{u} - I_x I_t) && (3.63)\\
I_x I_y u + (\alpha^2 + I_y^2)v &= (\alpha^2 \bar{v} - I_y I_t), && (3.64)
\end{aligned}
$$

or in matrix form as:

$$
\underbrace{\begin{bmatrix} (\alpha^2 + I_x^2) & I_x I_y \\ I_x I_y & (\alpha^2 + I_y^2) \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \alpha^2 \bar{u} - I_x I_t \\ \alpha^2 \bar{v} - I_y I_t \end{bmatrix}. \tag{3.65}
$$

For robust estimation, we use ψ_d function for data term and ψ_s for smoothness term. Thus, we are minimizing:

$$
\int\int \psi_d \left((I_x u + I_y v + I_t)^2 \right) + \alpha^2 \psi_s \left(|\nabla u|^2 + |\nabla v|^2 \right) \tag{3.66}
$$

In this case, the final matrix becomes:

$$
\underbrace{\begin{bmatrix} (\psi_s{}' * \alpha^2 + \psi_d{}' * I_x^2) & \psi_d{}' * I_x I_y \\ \psi_d{}' * I_x I_y & (\psi_s{}' * \alpha^2 + \psi_d{}' * I_y^2) \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \psi_s{}' * \alpha^2 \bar{u} - \psi_d{}' * I_x I_t \\ \psi_s{}' * \alpha^2 \bar{v} - \psi_d{}' * I_y I_t \end{bmatrix}. \tag{3.67}
$$

These linear systems were solved using SOR (Successive Over Relaxation) iterative method.

Figure 3.7 shows the computed 2D optical flow in NSERC datasets using the regularization method with intensity data only.

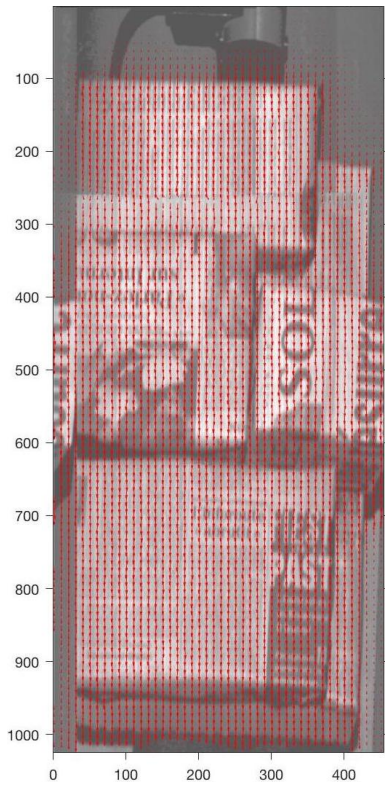


Figure 3.7: 2D optical flow using intensity data with the regularization method

3.2.2 Range Flow Regularization (Global_r)

Using range motion constraint equation:

$$Z_x u + Z_y v + Z_z w + Z_t = 0, \tag{3.68}$$

we want to regularize

$$f(x, y, t, u, v, w, u_x, u_y, u_t, u_z, v_x, v_y, v_t, v_z, w_x, w_y, w_t, w_z) =$$

$$(Z_x u + Z_y v + Z_z w + Z_t)^2 + \alpha^2(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2) \tag{3.69}$$

to compute (u, v, w) by minimizing:

$$\int\int\int\int f(x, y, t, u, v, w, u_x, u_y, u_z, u_t, v_x, v_y, v_z, v_t, w_x, w_y, w_z, w_t)\partial x\partial y\partial z\partial t \tag{3.70}$$

We are using 3D Euler-Lagrange equations where:

$$f_u = 2Z_x(Z_x u + Z_y v + Z_z w + Z_t) \tag{3.71}$$
$$f_v = 2Z_y(Z_x u + Z_y v + Z_z w + Z_t) \tag{3.72}$$
$$f_w = 2Z_z(Z_x u + Z_y v + Z_z w + Z_t) \tag{3.73}$$
$$f_{u_x} = 2\alpha^2 u_x \tag{3.74}$$

$$f_{u_y} = 2\alpha^2 u_y \tag{3.75}$$
$$f_{u_z} = 2\alpha^2 u_z \tag{3.76}$$
$$f_{u_t} = 2\alpha^2 u_t \tag{3.77}$$
$$f_{v_x} = 2\alpha^2 v_x \tag{3.78}$$
$$f_{v_y} = 2\alpha^2 v_y \tag{3.79}$$
$$f_{v_z} = 2\alpha^2 v_z \tag{3.80}$$
$$f_{v_t} = 2\alpha^2 v_t \tag{3.81}$$
$$f_{w_x} = 2\alpha^2 w_x \tag{3.82}$$
$$f_{w_y} = 2\alpha^2 w_y \tag{3.83}$$
$$f_{w_z} = 2\alpha^2 w_z \tag{3.84}$$
$$f_{w_t} = 2\alpha^2 w_t \tag{3.85}$$
$$\frac{df_{u_x}}{dx} = 2\alpha^2 u_{xx} \tag{3.86}$$
$$\frac{df_{u_y}}{dy} = 2\alpha^2 u_{yy} \tag{3.87}$$
$$\frac{df_{u_z}}{dz} = 2\alpha^2 u_{zz} \tag{3.88}$$
$$\frac{df_{u_t}}{dt} = 2\alpha^2 u_{tt} \tag{3.89}$$
$$\frac{df_{v_x}}{dx} = 2\alpha^2 v_{xx} \tag{3.90}$$
$$\frac{df_{v_y}}{dy} = 2\alpha^2 v_{yy} \tag{3.91}$$
$$\frac{df_{v_z}}{dz} = 2\alpha^2 v_{zz} \tag{3.92}$$
$$\frac{df_{v_t}}{dt} = 2\alpha^2 v_{tt} \tag{3.93}$$
$$\frac{df_{w_x}}{dx} = 2\alpha^2 w_{xx} \tag{3.94}$$
$$\frac{df_{w_y}}{dy} = 2\alpha^2 w_{yy} \tag{3.95}$$
$$\frac{df_{w_z}}{dz} = 2\alpha^2 w_{zz} \tag{3.96}$$
$$\frac{df_{w_t}}{dt} = 2\alpha^2 w_{tt}. \tag{3.97}$$

Since $\nabla^2 u = u_{xx} + u_{yy} + u_{zz} + u_{tt}$, $\nabla^2 v = v_{xx} + v_{yy} + v_{zz} + v_{tt}$ and $\nabla^2 w = w_{xx} + w_{yy} + w_{zz} + w_{tt}$ we can rewrite the Euler-Lagrange equations as:

$$Z_x^2 u + Z_x Z_y v + Z_x Z_z w + Z_x Z_t = \alpha^2 \nabla^2 u \tag{3.98}$$
$$Z_x Z_y u + Z_y^2 v + Z_y Z_z w + Z_y Z_t = \alpha^2 \nabla^2 v \tag{3.99}$$
$$Z_x Z_z u + Z_y Z_z v + Z_z^2 w + Z_z Z_t = \alpha^2 \nabla^2 w \tag{3.100}$$

(3.101)

Using $\nabla^2 u \approx \bar{u} - u$, $\nabla^2 v \approx \bar{v} - v$ and $\nabla^2 w \approx \bar{w} - w$ we can rewrite the Euler-Lagrange equations as:

$$(\alpha^2 + Z_x^2)u + Z_x Z_y v + Z_x Z_z w = \alpha^2 \bar{u} + Z_x Z_t \quad (3.102)$$

$$Z_x Z_y u + (\alpha^2 + Z_y^2)v + Z_y Z_z w = \alpha^2 \bar{v} + Z_y Z_t \quad (3.103)$$

$$Z_x Z_z u + Z_y Z_z v + (\alpha^2 + Z_z^2)w = \alpha^2 \bar{w} + Z_z Z_t. \quad (3.104)$$

for range data $Z_z = 1$ and $Z_{zz} = 0$. So, this system can be defined in a matrix form as:

$$\underbrace{\begin{bmatrix} (\alpha^2 + Z_x^2) & Z_x Z_y & Z_x \\ Z_x Z_y & (\alpha^2 + Z_y^2) & Z_y \\ Z_x & Z_y & (\alpha^2 + 1) \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \alpha^2 \bar{u} - Z_x Z_t \\ \alpha^2 \bar{v} - Z_y Z_t \\ \alpha^2 \bar{w} - Z_t \end{bmatrix}. \quad (3.105)$$

For robust estimation, we used ψ_d function for data term and ψ_s for smoothness term. Thus, we are minimizing:

$$\int\int\int\int \psi_d \left((Z_x u + Z_y v + Z_z w + Z_t)^2 \right) + \alpha^2 \psi_s \left(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2 \right) \partial x \partial y \partial z \partial t \quad (3.106)$$

The final matrix that needs to be solved becomes:

$$\underbrace{\begin{bmatrix} \psi_s' * \alpha^2 + \psi_d' * Z_x^2 & \psi_d' * Z_x Z_y & \psi_d' * Z_x \\ \psi_d' * Z_x Z_y & \psi_s' * \alpha^2 + \psi_d' * Z_y^2 & \psi_d' * Z_y \\ \psi_d' * Z_x & \psi_d' * Z_y & \psi_s' * \alpha^2 + \psi_d' * 1 \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \psi_s' * \alpha^2 \bar{u} - \psi_d' * Z_x Z_t \\ \psi_s' * \alpha^2 \bar{v} - \psi_d' * Z_y Z_t \\ \psi_s' * \alpha^2 \bar{w} - \psi_d' * Z_t \end{bmatrix}. \quad (3.107)$$

These linear systems are solved using SOR (Successive Over Relaxation) iterative method.

Figure 3.8 shows the computed 3D range flow in NSERC datasets using the regularization method with range data only.

3.2.3 Range Flow Regularization Using Intensity and Range Data (Global_ir)

It is possible to compute $\vec{v} = (u, v, w)$ using both intensity and range derivatives by combining the motion constraint equation with range constraint equation. While using same smoothing term as above. Consequently, we need to minimize:

$$f = (Z_x u + Z_y v + Z_z w + Z_t)^2 + \beta^2 (I_x u + I_y v + I_t)^2$$

$$+\alpha^2 (u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2) \quad (3.108)$$

The necessary derivatives for Euler-Lagrange equations are:

$$f_u = 2Z_x (Z_x u + Z_y v + w + Z_t) + 2\beta^2 I_x (I_x u + I_y v + I_t) \quad (3.109)$$

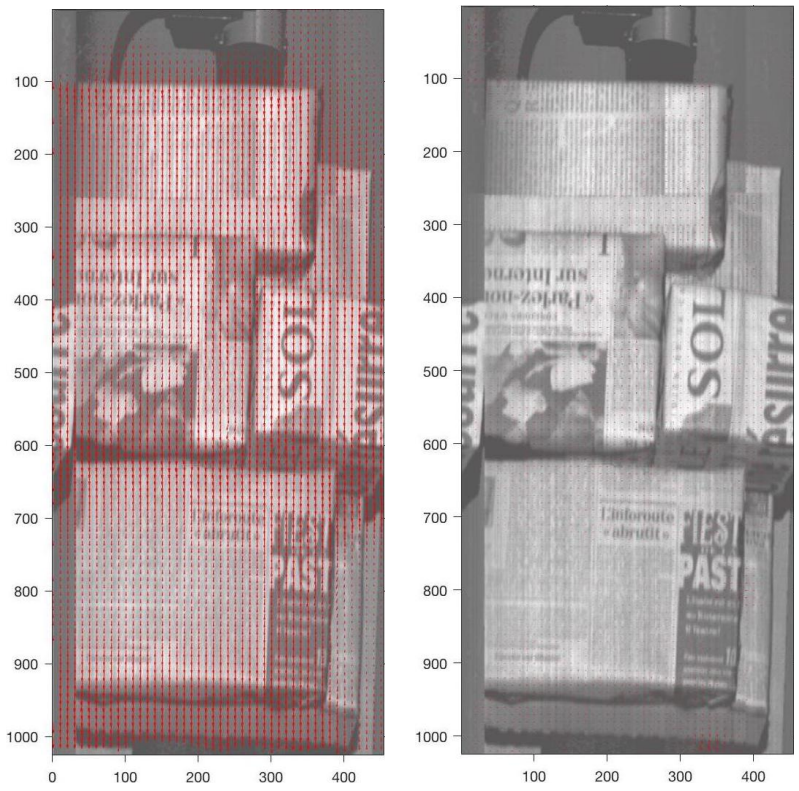


Figure 3.8: 3D range flow using range data with the direct regularization method (a) u,v components of 3D range flow, and (b) u,w components

$$f_v = 2Z_y(Z_x u + Z_y v + w + Z_t) + 2\beta^2 I_y(I_x u + I_y v + I_t) \quad (3.110)$$
$$f_w = 2(Z_x u + Z_y v + w + Z_t). \quad (3.111)$$

The other derivatives are the same as in Equations (3.74) to (3.97). We can write the Euler-Lagrange equations as:

$$Z_x(Z_x u + Z_y v + w + Z_t) + \beta^2 I_x(I_x u + I_y v + I_t) = \alpha^2 \nabla^2 u \quad (3.112)$$
$$Z_y(Z_x u + Z_y v + w + Z_t) + \beta^2 I_y(I_x u + I_y v + I_t) = \alpha^2 \nabla^2 v \quad (3.113)$$
$$(Z_x u + Z_y v + w + Z_t) = \alpha^2 \nabla^2 w. \quad (3.114)$$

Using $\nabla^2 u \approx \bar{u} - u$, $\nabla^2 v \approx \bar{v} - v$ and $\nabla^2 w \approx \bar{w} - w$ as above, we can rewrite the Euler-Lagrange equations as:

$$(\alpha^2 + Z_x^2 + \beta^2 I_x^2)u + (Z_x Z_y + \beta^2 I_x I_y)v + Z_x w = \alpha^2 \bar{u} + Z_x Z_t \quad (3.115)$$
$$(Z_x Z_y + \beta^2 I_x I_y)u + (\alpha^2 + Z_y^2 + \beta^2 I_y^2)v + Z_y w = \alpha^2 \bar{v} + Z_y Z_t \quad (3.116)$$
$$Z_x u + Z_y v + \alpha^2 w = \alpha^2 \bar{w} + Z_t. \quad (3.117)$$

In matrix form, this becomes:

$$\underbrace{\begin{bmatrix} \alpha^2 + Z_x^2 + \beta^2 I_x^2 & Z_x Z_y + \beta^2 I_x I_y & Z_x \\ Z_x Z_y + \beta^2 I_x I_y & \alpha^2 + Z_y^2 + \beta^2 I_y^2 & Z_y \\ Z_x & Z_y & \alpha^2 + 1 \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \alpha^2 \bar{u} - Z_x Z_t + \beta^2 I_x I_t \\ \alpha^2 \bar{v} - Z_y Z_t + \beta^2 I_y I_t \\ \alpha^2 \bar{w} - Z_t \end{bmatrix}. \tag{3.118}$$

When $\beta = 0$ we have standard range flow regularization.

For robust estimation, we used ψ_d function for data term and ψ_s for smoothness term. So, we are minimizing:

$$\int\int\int\int \psi_d \left((Z_x u + Z_y v + Z_z w + Z_t)^2 + \beta^2 (I_x u + I_y v + I_t)^2 \right) \alpha^2 \psi_s \left(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2 \right) \partial x \partial y \partial z \partial t \tag{3.119}$$

The final matrix that needs to be solved becomes:

$$\underbrace{\begin{bmatrix} \psi_s' * \alpha^2 + \psi_d' * (Z_x^2 + \beta^2 I_x^2) & \psi_d' * (Z_x Z_y + \beta^2 I_x I_y) & \psi_d' * Z_x \\ \psi_d' * (Z_x Z_y + \beta^2 I_x I_y) & \psi_s' * \alpha^2 + \psi_d' * (Z_y^2 + \beta^2 I_y^2) & \psi_d' * Z_y \\ \psi_d' * Z_x & \psi_d' * Z_y & \psi_s' * \alpha^2 + \psi_d' * 1 \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$= \begin{bmatrix} \psi_s' * \alpha^2 \bar{u} - \psi_d' * (Z_x Z_t + \beta^2 I_x I_t) \\ \psi_s' * \alpha^2 \bar{v} - \psi_d' * (Z_y Z_t + \beta^2 I_y I_t) \\ \psi_s' * \alpha^2 \bar{w} - \psi_d' * Z_t \end{bmatrix}. \tag{3.120}$$

This linear system was solved using SOR (Successive Over Relaxation) iterative method.

Figure 3.9 shows the computed 3D range flow in NSERC datasets using the regularization method when combining intensity and range data together.

3.2.4 Optical Flow by Indirect Regularization (Global_ind_i)

Optical flow $\vec{V} = (u, v)$ can be regularized and computed using least-square flow as initial values.

Given a $\vec{v}_f$ and $\vec{v}_n$ at each location (x, y) and the eigenvalues and eigenvectors, we can use a regularization calculation to produce dense smooth optical flow at each image location.

Since we measured velocity $\vec{V}_M$ using least square in the previous step, $\vec{V}_M$ can be either one of $\vec{v}_f$ (full velocity), $\vec{v}_v$ (normal velocity), or $\vec{v}_0$ (no velocity, set to (0,0,0)) stored in it, where the values of t_0 and t_1 (values of 0 or 1) indicate what is stored in $\vec{V}_M$. The table below shows the t values for the different types of velocities.

t_0	t_1	Type of $\vec{v}$
1	0	$\vec{v}_f$
0	1	$\vec{v}_n$
0	0	$\vec{v}_0$

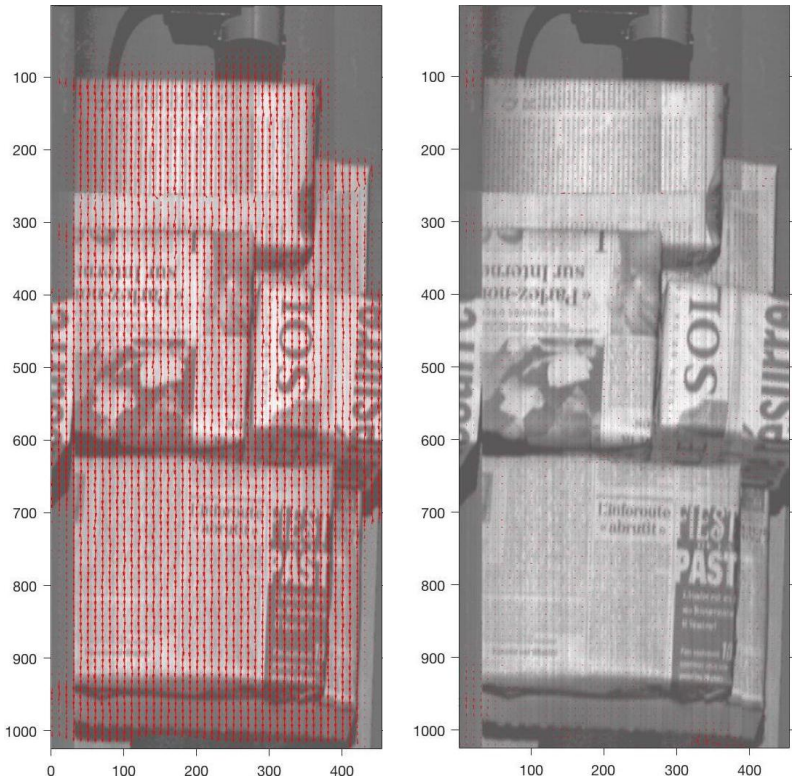


Figure 3.9: 3D range flow using intensity/range data with the direct regularization method (a) u,v components of 3D range flow, and (b) u,w components

That is, for $A\vec{V} = B$, $\vec{V}_M$ is one of:

$$\vec{v}_f = (\vec{V} \cdot \hat{e}_0)\hat{e}_0 \tag{3.121}$$

$$\vec{v}_n = (\vec{V} \cdot \hat{e}_1)\hat{e}_1 \tag{3.122}$$

We can write this as:

$$V_M = t_0(\vec{V} \cdot \hat{e}_0)\hat{e}_0 + t_1(\vec{V} \cdot \hat{e}_1)\hat{e}_1$$

Of course, $\vec{V}_M$ has 2 components, (U_M, V_M). Note that $\hat{e}_0$ and $\hat{e}_1$ are the eigenvectors of the matrix A and used to solve for $\vec{V}$ in equation systems $A\vec{V} = B$.

Therefore, to compute 2D optical flow $\vec{V} = (u, v)$, we want to regularize:

$$f(x, y, u, v, u_x, u_y, v_x, v_y) =$$

$$\int\int \|t_0(\vec{V} \cdot \hat{e}_0)\hat{e}_0 + t_1(\vec{V} \cdot \hat{e}_1)\hat{e}_1 - \vec{V}_M\|_2^2 + \alpha^2(u_x^2 + u_y^2 + v_x^2 + v_y^2) \tag{3.123}$$

The data term $\left[t_0(\vec{V} \cdot \hat{e}_0)\hat{e}_0 + t_1(\vec{V} \cdot \hat{e}_1)\hat{e}_1 - \vec{V}_M\right]$ can be written as $\left[B\vec{V} - \vec{V}_M\right]$ where:

$$B = t_0\hat{e_0}^T \cdot \hat{e}_0 + t_1\hat{e_1}^T \cdot \hat{e}_1$$

B is an idempotent matrix (i.e. $BB = B$).

Note that B is a 2×2 matrix. $B(1,:), B(2,:)$ are the two rows of B, while $B(:,1), B(:,2)$ are the two columns of B. Also, $(\vec{V} \cdot \hat{e}_0)\hat{e}_0$ can be written as:

$$((U,V) \cdot (e00, e01))(e00, e01) =$$

$$\begin{pmatrix} (Ue00 + Ve01)e00 \\ (Ue00 + Ve01)e01 \end{pmatrix} = \begin{pmatrix} (Ue00e00 + Ve01e00) \\ (Ue00e01 + Ve01e01) \end{pmatrix} \tag{3.124}$$

We can write this as:

$$(U,V)(e00, e01)^T \cdot (e00, e01) = (U,V)(\hat{e}_0^T \cdot \hat{e}_0) = \vec{V}(\hat{e}_0^T \cdot \hat{e}_0). \tag{3.125}$$

Similarly, $(\vec{V} \cdot \hat{e}_1)\hat{e}_1$ can be shown as equal to $\vec{V}(\hat{e}_1^T \cdot \hat{e}_1)$

We use the Euler-Lagrange equations to minimize this integral. We can compute derivative as follows:

$$\frac{\partial F(\vec{V})}{\partial \vec{V}} = 2B(\vec{V} - \vec{V}_M). \tag{3.126}$$

The necessary derivatives are:

$$F_u = 2B(1,:) \cdot (\vec{V} - \vec{V}_M) \tag{3.127}$$

$$F_v = 2B(2,:) \cdot (\vec{V} - \vec{V}_M) \tag{3.128}$$

Or:

$$\begin{pmatrix} f_u \\ f_v \end{pmatrix} = 2B(\vec{V} - \vec{V}_M) \tag{3.129}$$

The other derivatives are the same as in Equations (3.54) to (3.60).

We can rewrite the Euler-Lagrange equations as:

$$B(\vec{V} - \vec{V}_M) = \alpha^2 \nabla^2 \vec{V} \tag{3.130}$$

Using $\nabla^2 \vec{V} \approx \vec{V}_{ave} - \vec{V}$, where $\vec{V}_{ave} = (\bar{u}, \bar{v})$, we can rewrite the Euler-Lagrange equations as:

$$B(\vec{V} - \vec{V}_M) = \alpha^2 (\vec{V}_{ave} - \vec{V}) \tag{3.131}$$

or, after re-arrangement, as:

$$\underbrace{(B + \alpha^2 I)}_{A} \vec{V} = \alpha^2 \vec{V}_{ave} + B\vec{V}_M. \tag{3.132}$$

For robust estimation, we used ψ_d function for the data term and ψ_s for the smoothness term. Thus, we are minimizing:

$$\int\int\int \psi_d \left((B\vec{V} - \vec{V}_M)^2 \right) + \alpha^2 \psi_s \left(|\nabla u|^2 + |\nabla v|^2 \right) \partial x \partial y \partial t \tag{3.133}$$

The final matrix that needs to be solved becomes:

$$\underbrace{(\psi_d' * B + \psi_s' * \alpha^2 I)}_{A} \vec{V} = \psi_s' * \alpha^2 \vec{V}_{ave} + \psi_d' * (B\vec{V}_M). \tag{3.134}$$

Figure 3.10 shows the computed 2D optical flow in NSERC datasets using the indirect regularization method with intensity data only.

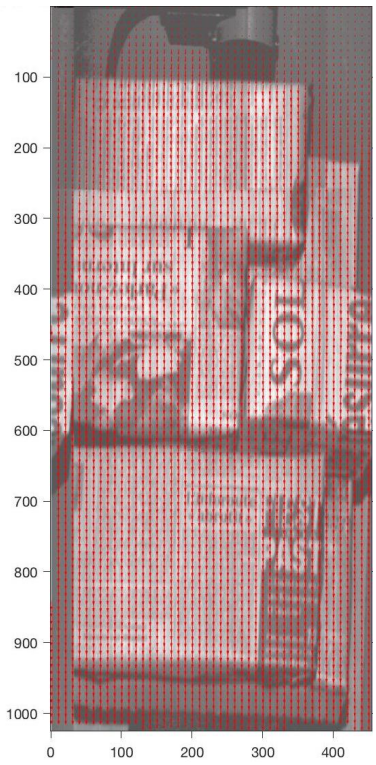


Figure 3.10: 2D optical flow using intensity data with the indirect regularization method

3.2.5 Range Flow by Indirect Regularization (Global_ind_r)

As described in [89] and [98], range flow $\vec{V} = (u, v, w)$ can be regularized and computed using least-square flow as initial values.

Given a $\vec{v}_f$, $\vec{v}_l$ or $\vec{v}_p$ at each location (x, y) and the eigenvalues and eigenvectors, we can use a regularization calculation to produce dense smooth range flow at each image location.

Since we measured velocity $\vec{V}_M$ using least square in the previous step, $\vec{V}_M$ can be either one of $\vec{v}_f$ (full velocity), $\vec{v}_l$ (line normal velocity), $\vec{v}_p$ (plane velocity) or $\vec{v}_0$ (no velocity, set to (0,0,0)) stored in it, where the values of t_0, t_1 and t_2 (values of 0 or 1) indicate what is stored in $\vec{V}_M$. The table below shows the t values for the different types of velocities.

t_0	t_1	t_2	Type of $\vec{v}$
1	1	1	$\vec{v}_f$
0	1	1	$\vec{v}_l$
0	0	1	$\vec{v}_p$
0	0	0	$\vec{v}_0$

That is, for $A\vec{V} = B$, $\vec{V}_M$ is one of:

$$\vec{v}_f = (\vec{V} \cdot \hat{e}_0)\hat{e}_0 + (\vec{V} \cdot \hat{e}_1)\hat{e}_1 + (\vec{V} \cdot \hat{e}_2)\hat{e}_2 \tag{3.135}$$

$$\vec{v}_l = (\vec{V} \cdot \hat{e}_1)\hat{e}_1 + (\vec{V} \cdot \hat{e}_2)\hat{e}_2 \tag{3.136}$$

$$\vec{v}_p = (\vec{V} \cdot \hat{e}_2)\hat{e}_2. \tag{3.137}$$

We can write this as:

$$V_M = t_0(\vec{V} \cdot \hat{e}_0)\hat{e}_0 + t_1(\vec{V} \cdot \hat{e}_1)\hat{e}_1 + t_2(\vec{V} \cdot \hat{e}_2)\hat{e}_2$$

Of course, $\vec{V}_M$ has 3 components, (U_M, V_M, W_M). Note that $\hat{e}_0$, $\hat{e}_1$ and $\hat{e}_2$ are the eigenvectors of the matrix A and used to solve for $\vec{V}$ in equation systems $A\vec{V} = B$.

Therefore, to compute 3D range flow $\vec{V} = (u, v, w)$, we want to regularize:

$$f(x, y, t, u, v, w, u_x, u_y, u_t, u_z, v_x, v_y, v_z, v_t, w_x, w_y, w_z, w_t) =$$

$$\int\int\int \|t_0(\vec{V} \cdot \hat{e}_0)\hat{e}_0 + t_1(\vec{V} \cdot \hat{e}_1)\hat{e}_1 + t_2(\vec{V} \cdot \hat{e}_2)\hat{e}_2 - \vec{V}_M\|_2^2$$

$$+\alpha^2(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2) \tag{3.138}$$

The data term $\left[t_0(\vec{V} \cdot \hat{e}_0)\hat{e}_0 + t_1(\vec{V} \cdot \hat{e}_1)\hat{e}_1 + t_2(\vec{V} \cdot \hat{e}_2)\hat{e}_2 - \vec{V}_M\right]$ can be written as $\left[B\vec{V} - \vec{V}_M\right]$ where:

$$B = t_0\hat{e_0}^T \cdot \hat{e}_0 + t_1\hat{e_1}^T \cdot \hat{e}_1 + t_2\hat{e_2}^T \cdot \hat{e}_2$$

B is an idempotent matrix (i.e. $BB = B$).

Note that B is a 3×3 matrix. $B(1,:), B(2,:), B(3,:)$ are the three rows of B, while $B(:,1), B(:,2), B(:,3)$ are the three columns of B. Also, $(\vec{V} \cdot \hat{e}_0)\hat{e}_0$ can be written as:

$$((U, V, W) \cdot (e00, e01, e02))(e00, e01, e02) =$$

$$\begin{pmatrix} (Ue00 + Ve01 + We02)e00 \\ (Ue00 + Ve01 + We02)e01 \\ (Ue00 + Ve01 + We02)e02 \end{pmatrix} = \begin{pmatrix} (Ue00e00 + Ve01e00 + We02e00) \\ (Ue00e01 + Ve01e01 + We02e01) \\ (Ue00e02 + Ve01e02 + We02e02), \end{pmatrix} \tag{3.139}$$

We can write this as:

$$(U, V, W)(e00, e01, e02)^T \cdot (e00, e01, e02) = (U, V, W)(\hat{e}_0^T \cdot \hat{e}_0) = \vec{V}(\hat{e}_0^T \cdot \hat{e}_0). \tag{3.140}$$

Similarly, $(\vec{V} \cdot \hat{e}_1)\hat{e}_1$ and $(\vec{V} \cdot \hat{e}_2)\hat{e}_2$ can be shown as equal to $\vec{V}(\hat{e}_1^T \cdot \hat{e}_1)$ and $\vec{V}(\hat{e}_2^T \cdot \hat{e}_2)$

We use the Euler-Lagrange equations to minimize this integral. We can compute derivative as follows:

$$\frac{\partial F(\vec{V})}{\partial \vec{V}} = 2B(\vec{V} - \vec{V}_M). \tag{3.141}$$

The necessary derivatives are:

$$\begin{aligned} F_U &= 2B(1,:) \cdot (\vec{V} - \vec{V}_M) && (3.142)\\ F_V &= 2B(2,:) \cdot (\vec{V} - \vec{V}_M) && (3.143)\\ F_W &= 2B(3,:) \cdot (\vec{V} - \vec{V}_M) && (3.144) \end{aligned}$$

Or:

$$\begin{pmatrix} f_u \\ f_v \\ f_w \end{pmatrix} = 2B(\vec{V} - \vec{V}_M) \tag{3.145}$$

The other derivatives are the same as in Equations (3.74) to (3.97).

We can rewrite the Euler-Lagrange equations as:

$$B(\vec{V} - \vec{V}_M) = \alpha^2 \nabla^2 \vec{V} \tag{3.146}$$

Using $\nabla^2 \vec{V} \approx \vec{V}_{ave} - \vec{V}$, where $\vec{V}_{ave} = (\bar{u}, \bar{v}, \bar{w})$, we can rewrite the Euler-Lagrange equations as:

$$B(\vec{V} - \vec{V}_M) = \alpha^2 (\vec{V}_{ave} - \vec{V}) \tag{3.147}$$

or, after re-arrangement, as:

$$\underbrace{(B + \alpha^2 I)}_{A} \vec{V} = \alpha^2 \vec{V}_{ave} + B\vec{V}_M. \tag{3.148}$$

For robust estimation, we used ψ_d function for the data term and ψ_s for the smoothness term. Thus, we are minimizing:

$$\int\int\int\int \psi_d \left((B\vec{V} - \vec{V}_M)^2 \right) + \alpha^2 \psi_s \left(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2 \right) \partial x \partial y \partial z \partial t \tag{3.149}$$

The final matrix that needs to be solved becomes:

$$\underbrace{(\psi_d' * B + \psi_s' * \alpha^2 I)}_{A} \vec{V} = \psi_s' * \alpha^2 \vec{V}_{ave} + \psi_d' * (B\vec{V}_M). \tag{3.150}$$

Figure 3.11 shows the computed 3D range flow in NSERC datasets using the indirect regularization method with range data only.

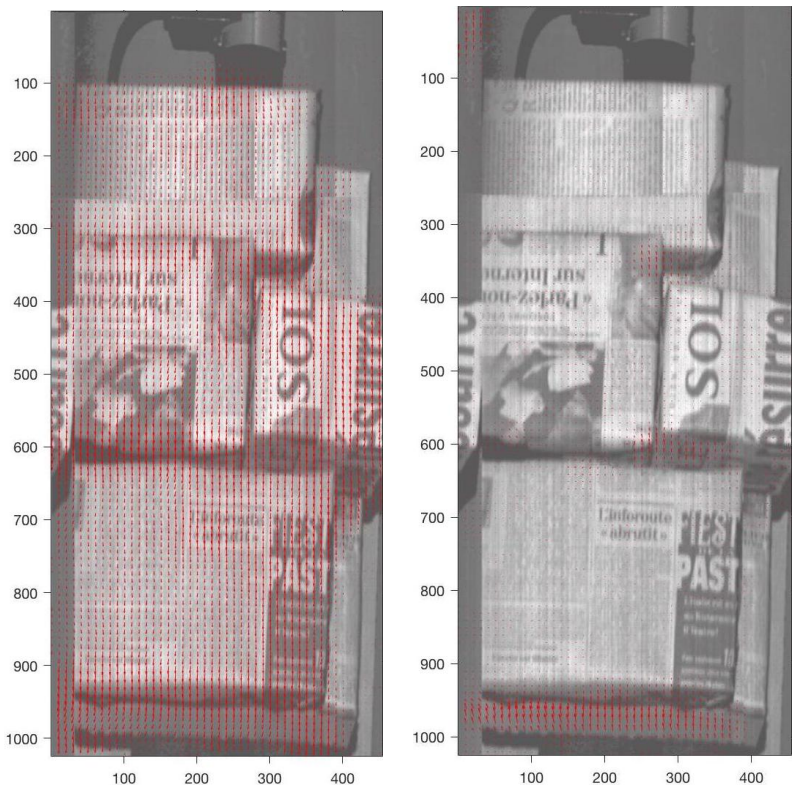


Figure 3.11: 3D range flow using range data with the indirect regularization method (a) u,v components of 3D range flow, and (b) u,w components

3.2.6 Range Flow by Indirect Regularization Using Intensity and Range data (Global_ind_ir)

It is possible to compute 3D range flow $\vec{V} = (u, v, w)$ by combining intensity and range data in the indirect regularization approach. The initial values for this regularization are the output of the least-square method that combine intensity and range data.

Given a $\vec{v}_f$, $\vec{v}_l$ or $\vec{v}_p$ at each location (x, y) and the eigenvalues and eigenvectors, we can use a regularization calculation to produce dense smooth range flow at each image location.

Following the previous approach, to compute 3D range flow $\vec{V} = (u, v, w)$, we want to regularize:

$$f(x, y, t, u, v, w, u_x, u_y, u_t, u_z, v_x, v_y, v_z, v_t, w_x, w_y, w_z, w_t) =$$
$$\int\int\int \|t_0(\vec{V}\cdot\hat{e}_0)\hat{e}_0 + t_1(\vec{V}\cdot\hat{e}_1)\hat{e}_1 + t_2(\vec{V}\cdot\hat{e}_2)\hat{e}_2 - \vec{V}_M\|_2^2$$
$$+\alpha^2(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2) \qquad (3.151)$$

The data term $\left[t_0(\vec{V}\cdot\hat{e}_0)\hat{e}_0 + t_1(\vec{V}\cdot\hat{e}_1)\hat{e}_1 + t_2(\vec{V}\cdot\hat{e}_2)\hat{e}_2 - \vec{V}_M\right]$ can be written as $\left[B\vec{V} - \vec{V}_M\right]$ where:

$$B = t_0\hat{e_0}^T\cdot\hat{e}_0 + t_1\hat{e_1}^T\cdot\hat{e}_1 + t_2\hat{e_2}^T\cdot\hat{e}_2$$

B is an idempotent matrix constructed using the eigenvectors that computed using the combination of intensity and range data.

$\vec{V}_M$ is the output (full, line, and plane) flow that was computed using the combination of intensity and range data in the least square method.

The Euler-Lagrange equation is defined as:

$$\underbrace{(B + \alpha^2 I)}_{A} \vec{V} = \alpha^2 \vec{V}_{ave} + B\vec{V}_M. \tag{3.152}$$

For robust estimation, we are using ψ_d function for data term and ψ_s for smoothness term. So, we are minimizing:

$$\int\int\int\int \psi_d \left((B\vec{V} - \vec{V}_M)^2\right) + \alpha^2 \psi_s \left(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2\right) \partial x \partial y \partial z \partial t \tag{3.153}$$

The final matrix that needs to be solved becomes:

$$\underbrace{(\psi_d' * B + \psi_s' * \alpha^2 I)}_{A} \vec{V} = \psi_s' * \alpha^2 \vec{V}_{ave} + \psi_d' * (B\vec{V}_M). \tag{3.154}$$

Figure 3.12 shows the computed 3D range flow in NSERC datasets using the indirect regularization method by combining intensity and range data together.

3.3 Combined Local and Global Approach (CLG):

3.3.1 Optical Flow by CLG (CLG_i)

Bruhn et al. in [4, 5] juxtaposed the smoothness role required for local and global methods. They came up with a new approach that combines the advantages of local and global estimation algorithms. In the local method, Lucas and Kanade [1] assume the optical flow is constant a neighbourhood size p to solve the aperture problem. We can find (u, v) at a location from a weighted least-square fit by minimizing the function:

$$E_{LK}(u, v) = K_\rho \left((I_x u + I_y v + I_t)^2\right). \tag{3.155}$$

Here K_ρ is the standard deviation of the Gaussian used to smooth the images. In matrix form:

$$\begin{bmatrix} K_\rho * I_x^2 & K_\rho * I_x I_y \\ K_\rho * I_x I_y & K_\rho * I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} K_\rho * I_x I_t \\ K_\rho * I_y I_t \end{bmatrix}. \tag{3.156}$$

To determine the dense flow, we need to use regularization frameworks like Horn and Shunck's approach [2].

$$E_{HS}(u, v) = (I_x u + I_y v + I_t)^2 + \alpha^2 |\nabla u|^2 + |\nabla v|^2 \tag{3.157}$$

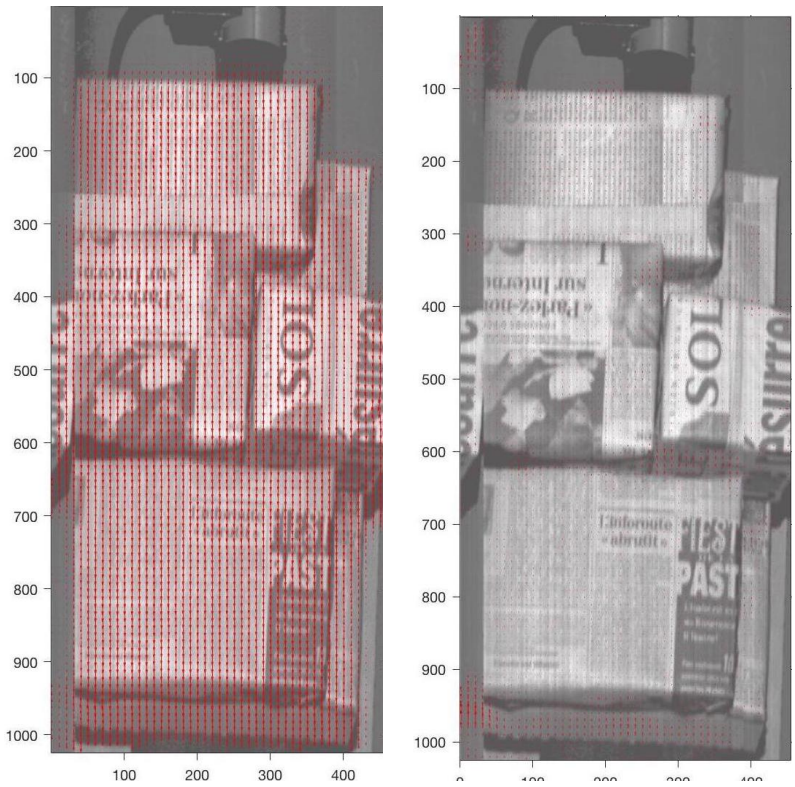


Figure 3.12: 3D range flow using intensity/range data with indirect regularization method (a) u,v components of 3D range flow, and (b) u,w components

To estimate (u, v) by combining local and global approaches, we need to minimize the following function:

$$E_{CLG}(u,v) = \int_{\Omega(x,y,t)} \psi_d \left(K_\rho * \left((I_x u + I_y v + I_t)^2 \right) \right) + \alpha^2 \psi_s \left(|\nabla u|^2 + |\nabla v|^2 \right) dxdydt. \tag{3.158}$$

Where ψ is a robust function, Bruhn defined it as:

$$\psi(s^2) = 2\beta^2 \sqrt{1 + \frac{s^2}{\beta^2}}, \tag{3.159}$$

Following the similar Euler-Lagrange equation and derivatives for data term and smoothing part in Section 3.2.1 for 2D optical flow regularization, we can formulate the matrix as:

$$\underbrace{\begin{bmatrix} \alpha^2 + K_\rho * I_x^2 & K_\rho * I_x I_y \\ K_\rho * I_x I_y & \alpha^2 + K_\rho * I_y^2 \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \alpha^2 \bar{u} - K_\rho * I_x I_t \\ \alpha^2 \bar{v} - K_\rho * I_y I_t \end{bmatrix}. \tag{3.160}$$

Using robust estimation, the matrix becomes:

$$\underbrace{\begin{bmatrix} \psi_s' * \alpha^2 + \psi_d' * (K_\rho * I_x^2) & \psi_d' * (K_\rho * I_x I_y) \\ \psi_d' * (K_\rho * I_x I_y) & \psi_s' * \alpha^2 + \psi_d' * (K_\rho * I_y^2) \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \psi_s' * \alpha^2 \bar{u} - \psi_d' * (K_\rho * I_x I_t) \\ \psi_s' * \alpha^2 \bar{v} - \psi_d' * (K_\rho * I_y I_t) \end{bmatrix}. \tag{3.161}$$

These linear systems are solved using iterative solutions like SOR (Successive Over Relaxation) method.

Figure 3.13 shows the computed 2D optical flow in NSERC datasets using CLG method with intensity data only.

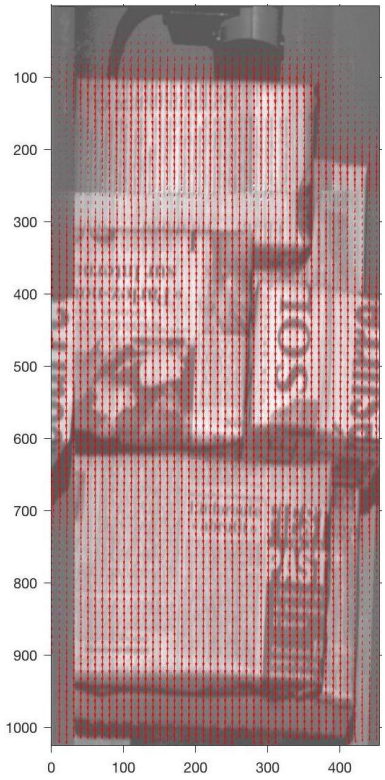


Figure 3.13: 2D optical flow using intensity data with CLG method

3.3.2 Range Flow CLG (CLG_r)

To compute range flow $\vec{v} = (u, v, w)$ using the CLG algorithm, we use a range flow constraint equation.

$$Z_x u + Z_y v + Z_z w + Z_t = 0, \tag{3.162}$$

For the local method, range flow can be defined as:

$$E_{LK}(u, v, w) = K_\rho \left((Z_x u + Z_y v + w + Z_t)^2 \right) \tag{3.163}$$

Here K_ρ is the standard deviation of the Gaussian used to smooth the images. In a matrix form, we use the following structure tensor:

$$\begin{bmatrix} K_\rho * Z_x^2 & K_\rho * Z_x Z_y & K_\rho * Z_x \\ K_\rho * Z_x Z_y & K_\rho * Z_y^2 & K_\rho * Z_y \\ K_\rho * Z_x & K_\rho * Z_y & 1 \\ K_\rho * Z_x Z_t & K_\rho * Z_y Z_t & K_\rho * Z_t \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} K_\rho * Z_x Z_t \\ K_\rho * Z_y Z_t \\ K_\rho * Z_t \\ K_\rho * Z_t^2 \end{bmatrix} \tag{3.164}$$

To determine dense flow, we use extend Horn and Shunck's [2] approach to regularization frameworks.

$$E_{HS}(u, v, w) = (Z_x u + Z_y v + w + Z_t)^2 + \alpha^2 |\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2 \tag{3.165}$$

To estimate (u, v, w) by combining local and global approaches, we need to minimize the following function:

$$E_{CLG}(u, v, w) = \int_{\Omega(x,y,z,t)} \psi_d \left(K_\rho * \left((Z_x u + Z_y v + w + Z_t)^2 \right) \right) + \alpha^2 \psi_s \left(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2 \right) dx dy dt. \tag{3.166}$$

Where ψ is the robust function, Bruhn defined it as:

$$\psi(s^2) = 2\beta^2 \sqrt{1 + \frac{s^2}{\beta^2}} \tag{3.167}$$

Following the similar Euler-Lagrange equation and derivatives for data term and smoothing part in Section 3.2.2 for 3D range flow regularization, we can formulate the matrix as:

$$\underbrace{\begin{bmatrix} \alpha^2 + K_\rho * Z_x^2 & K_\rho * Z_x Z_y & K_\rho * Z_x \\ K_\rho * Z_x Z_y & \alpha^2 + K_\rho * Z_y^2 & K_\rho * Z_y \\ K_\rho * Z_x & K_\rho * Z_y & \alpha^2 + K_\rho * 1 \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \alpha^2 \bar{u} - K_\rho * Z_x Z_t \\ \alpha^2 \bar{v} - K_\rho * Z_y Z_t \\ \alpha^2 \bar{w} - K_\rho * Z_t \end{bmatrix}. \tag{3.168}$$

Using robust estimation, the matrix became:

$$\underbrace{\begin{bmatrix} \psi_s' * \alpha^2 + \psi_d' * (K_\rho * Z_x^2) & \psi_d' * (K_\rho * Z_x Z_y) & \psi_d' * (K_\rho * Z_x) \\ \psi_d' * (K_\rho * Z_x Z_y) & \psi_s' * \alpha^2 + \psi_d' * (K_\rho * Z_y^2) & \psi_d' * (K_\rho * Z_y) \\ \psi_d' * (K_\rho * Z_x) & \psi_d' * (K_\rho * Z_y) & \psi_s' * \alpha^2 + \psi_d' * (K_\rho * 1) \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$= \begin{bmatrix} \psi_s' * \alpha^2 \bar{u} - \psi_d' * (K_\rho * Z_x Z_t) \\ \psi_s' * \alpha^2 \bar{v} - \psi_d' * (K_\rho * Z_y) Z_t \\ \psi_s' * \alpha^2 \bar{w} - \psi_d' * (K_\rho * Z_t) \end{bmatrix}. \tag{3.169}$$

These linear systems are solved using iterative solutions like SOR (Successive Over Relaxation) method.

Figure 3.14 shows the computed 3D range flow in NSERC datasets using the CLG method when using range data only.

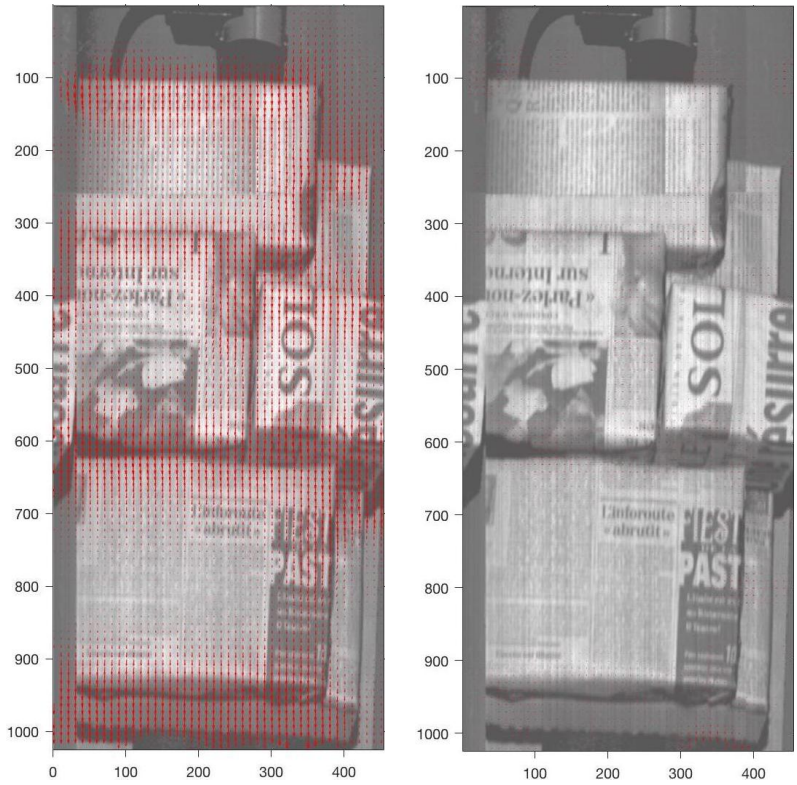


Figure 3.14: 3D range flow using range data with CLG method (a) u,v components of 3D range flow, and (b) u,w components

3.3.3 Range Flow by CLG Using Intensity and Range Data (CLG_ir)

Computing range flow (u, v, w) by combining intensity and range data in this method requires minimizing the following function.

$$E_{CLG}(u,v,w) = \int_{\Omega(x,y,z,t)} \psi_d \left(K_\rho * \left((Z_x u + Z_y v + w + Z_t)^2 \right) + K_\rho * \left((I_x u + I_y v + I_t)^2 \right) \right) + \alpha^2 \psi_s \left(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2 \right) dx dy dt. \tag{3.170}$$

The structure tensor in this case is:

$$\begin{bmatrix} K_\rho * (Z_x^2 + I_x^2) & K_\rho * (Z_x Z_y + I_x I_y) & K_\rho * (Z_x) \\ K_\rho * (Z_x Z_y + I_x I_y) & K_\rho * (Z_y^2 + I_y^2) & K_\rho * (Z_y) \\ K_\rho * (Z_x) & K_\rho * (Z_y) & K_\rho * 1 \\ K_\rho * (Z_x Z_t + I_x I_t) & K_\rho * (Z_y Z_t + I_y I_t) & K_\rho * (Z_t) \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} K_\rho * (Z_x Z_t + I_x I_t) \\ K_\rho * (Z_y Z_t + I_y I_t) \\ K_\rho * (Z_t) \\ K_\rho * (Z_t^2) \end{bmatrix}. \tag{3.171}$$

By applying similar Euler-Lagrange derivatives in a data term and smoothing part in section 3.2.3, we can formulate the matrix as:

$$\underbrace{\begin{bmatrix} \alpha^2 + K_\rho * (Z_x^2 + \beta^2 I_x^2) & K_\rho * (Z_x Z_y + \beta^2 I_x I_y) & K_\rho * (Z_x) \\ K_\rho * (Z_x Z_y + \beta^2 I_x I_y) & \alpha^2 + K_\rho * (Z_y^2 + \beta^2 I_y^2) & K_\rho * (Z_y) \\ K_\rho * (Z_x) & K_\rho * (Z_y) & \alpha^2 + K_\rho * 1 \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \alpha^2 \bar{u} - K_\rho * (Z_x Z_t + \beta^2 I_x I_t) \\ \alpha^2 \bar{v} - K_\rho * (Z_y Z_t + \beta^2 I_y I_t) \\ \alpha^2 \bar{w} - K_\rho * (Z_t) \end{bmatrix}. \tag{3.172}$$

Using robust estimation, the matrix became:

$$\underbrace{\begin{bmatrix} \psi_s' * \alpha^2 + \psi_d' * (K_\rho * (Z_x^2 + \beta^2 I_x^2)) & \psi_d' * (K_\rho * (Z_x Z_y + \beta^2 I_x I_y)) & \psi_d' * (K_\rho * (Z_x)) \\ \psi_d' * (K_\rho * (Z_x Z_y + \beta^2 I_x I_y)) & \psi_s' * \alpha^2 + \psi_d' * (K_\rho * (Z_y^2 + \beta^2 I_y^2)) & \psi_d' * (K_\rho * (Z_y)) \\ \psi_d' * (K_\rho * (Z_x)) & \psi_d' * (K_\rho * (Z_y)) & \psi_s' * \alpha^2 + \psi_d' * (K_\rho * 1) \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$= \begin{bmatrix} \psi_s' * \alpha^2 \bar{u} - \psi_d' * (K_\rho * (Z_x Z_t + \beta^2 I_x I_t)) \\ \psi_s' * \alpha^2 \bar{v} - \psi_d' * (K_\rho * (Z_y Z_t + \beta^2 I_y I_t)) \\ \psi_s' * \alpha^2 \bar{w} - \psi_d' * (K_\rho * (Z_t)) \end{bmatrix}. \tag{3.173}$$

These linear systems are solved using iterative method SOR (Successive Over Relaxation).

Figure 3.15 shows the computed 3D range flow in NSERC datasets using the CLG method when combining intensity and range data.

3.4 Brox et al. Method

3.4.1 Optical Flow by Brox et al. (Brox_i)

The Brox et al. [6] method computes optical flow $\vec{v} = (u, v)$ using the optical flow constraint equation:

$$I_x u + I_y v + I_t \quad = \quad 0. \tag{3.174}$$

This model adds the gradient constancy assumption. They assume that the gradient of the image grey values does not vary due to displacement. This supports the greyvalue constancy assumption. We have:

$$\nabla I(x, y, t) \quad = \quad \nabla I(x + u, y + v, t + 1). \tag{3.175}$$

Let $\mathbf{x} := (x, y, t)^\top$ and $\mathbf{w} := (u, v, 1)^\top$. Then, we measured the energy of the global deviations from the grey value constancy assumption and the gradient constancy assumption as:

$$E_{Data}(u, v) = \int \psi_d(|I(x + w) - I(x)|^2 + \gamma |\nabla I(x + w) - \nabla I(x)|^2) dx. \tag{3.176}$$

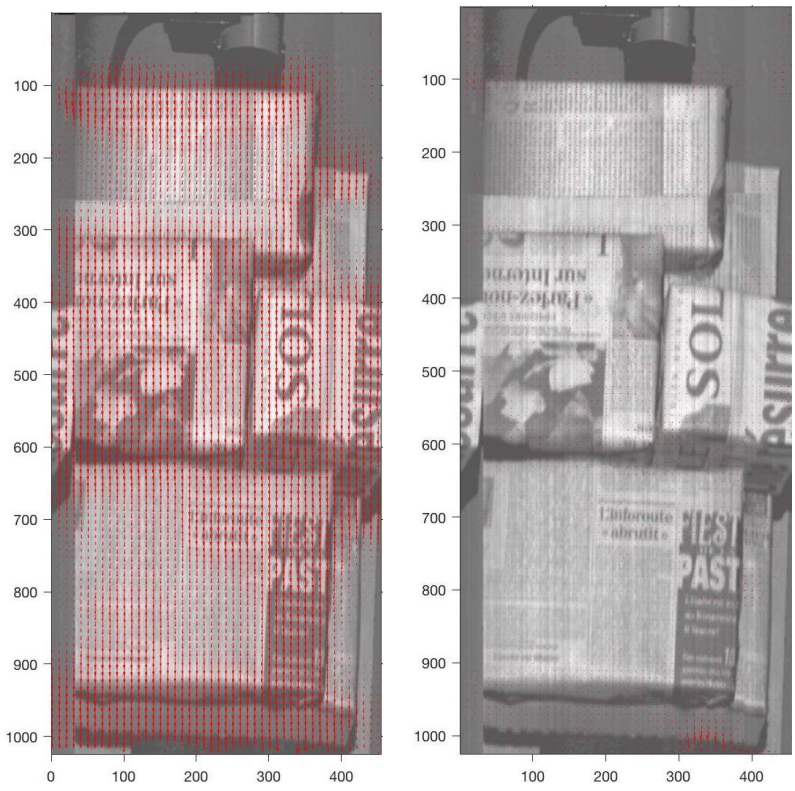


Figure 3.15: 3D range flow using intensity/range data with CLG method (a) u,v components of 3D range flow, and (b) u,w components

While the smoothness is expressed as:

$$E_{smooth}(u, v) = \int \psi_s(|\nabla_3 u|^2 + |\nabla_3 v|^2) dx, \tag{3.177}$$

The function ψ is a robust function, as defined by Brox:

$$\psi(s^2) = \sqrt{s^2 + \epsilon^2} \tag{3.178}$$

The small positive constant ϵ keeps $\psi(s^2)$ convex, which helps the overall minimization process. Brox et al. chose $\epsilon = 0.0001$.

The total energy is the weighted sum between the data term and the smoothness term

$$E(u, v) = E_{Data} + \alpha E_{smooth}, \tag{3.179}$$

We defined the Euler-Lagrange derivatives for Brox et al.'s optical flow energy function (3.179) using the following abbreviations:

$$I_x = \partial_x I(x + w), \tag{3.180}$$

$$
\begin{aligned}
I_y &= \partial_y I(x+w), && (3.181)\\
I_D &= I(x+w) - I(x), && (3.182)\\
I_{xx} &= \partial_{xx} I(x+w), && (3.183)\\
I_{xy} &= \partial_{xy} I(x+w), && (3.184)\\
I_{yy} &= \partial_{yy} I(x+w), && (3.185)\\
I_{xD} &= \partial_x I(x+w) - \partial_x I(x), && (3.186)\\
I_{yD} &= \partial_y I(x+w) - \partial_y I(x). && (3.187)
\end{aligned}
$$

Consequently, the energy function to be minimized is:

$$
\begin{aligned}
f(x,y,u,v,u_x,u_y,v_x,v_y) &= \psi_d\left(|I(x+w)-I(x)|^2+\gamma(|\nabla I(x+w)-\nabla I(x)|^2\right)+\alpha\psi_s\left(|\nabla_3 u|^2+|\nabla_3 v|^2\right)\\
&= \psi_d\left(I_D^2+\gamma(I_{xD}^2+I_{yD}^2)\right)+\alpha\psi_s\left(u_x^2+u_y^2+v_x^2+v_y^2\right). \qquad (3.188)
\end{aligned}
$$

Thus, the required derivatives are:

$$
\begin{aligned}
f_u &= \psi_d'\left(I_D^2+\gamma(I_{xD}^2+I_{yD}^2)\right)\left(I_D I_x+\gamma(I_{xD}I_{xx}+I_{yD}I_{xy})\right), && (3.189)\\
f_v &= \psi_d'\left(I_D^2+\gamma(I_{xD}^2+I_{yD}^2)\right)\left(I_D I_y+\gamma(I_{xD}I_{xy}+I_{yD}I_{yy})\right), && (3.190)
\end{aligned}
$$

We can now derive the smoothness derivative:

$$
\begin{aligned}
f_{u_x} &= \alpha\,\psi_s'\left(u_x^2+u_y^2+v_x^2+v_y^2\right)2u_x && (3.191)\\
f_{u_y} &= \alpha\,\psi_s'\left(u_x^2+u_y^2+v_x^2+v_y^2\right)2u_y && (3.192)\\
f_{v_x} &= \alpha\,\psi_s'\left(u_x^2+u_y^2+v_x^2+v_y^2\right)2v_x && (3.193)\\
f_{v_y} &= \alpha\,\psi_s'\left(u_x^2+u_y^2+v_x^2+v_y^2\right)2v_y && (3.194)\\
\frac{df_{u_x}}{dx} &= 2\alpha\,\psi_s'\left(u_x^2+u_y^2+v_x^2+v_y^2\right)u_{xx} && (3.195)\\
\frac{df_{u_y}}{dy} &= 2\alpha\,\psi_s'\left(u_x^2+u_y^2+v_x^2+v_y^2\right)u_{yy} && (3.196)\\
\frac{df_{v_x}}{dx} &= 2\alpha\,\psi_s'\left(u_x^2+u_y^2+v_x^2+v_y^2\right)v_{xx} && (3.197)\\
\frac{df_{v_y}}{dy} &= 2\alpha\,\psi_s'\left(u_x^2+u_y^2+v_x^2+v_y^2\right)v_{yy}. && (3.198)
\end{aligned}
$$

$\mathbf{Div}(\nabla_3 u) = u_{xx}+u_{yy}$, and $\mathbf{Div}(\nabla_3 v) = v_{xx}+v_{yy}$. so:

$$
\begin{aligned}
\frac{df_{u_x}}{dx}+\frac{df_{u_y}}{dy} &= \alpha\,\mathbf{Div}\,\nabla_3 u\,\psi_s'\left(|\nabla_3 u|^2+|\nabla_3 v|^2\right) && (3.199)\\
\frac{df_{v_x}}{dx}+\frac{df_{v_y}}{dy} &= \alpha\,\mathbf{Div}\,\nabla_3 v\,\psi_s'\left(|\nabla_3 u|^2+|\nabla_3 v|^2\right) && (3.200)
\end{aligned}
$$

Therefore, the final Euler-Lagrange equations as per Brox et al. are:

$$
\psi_d'\left(I_D^2+\gamma(I_{xD}^2+I_{yD}^2)\right)\left(I_x I_D+\gamma(I_{xx}I_{xD}+I_{xy}I_{yD})\right)-\alpha\,\mathbf{Div}\,\psi_s'\left(|\nabla_3 u|^2+|\nabla_3 v|^2\right)\nabla_3 u = 0 \qquad (3.201)
$$

$$\psi_d' \left(I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2)\right)\left(I_y I_D + \gamma(I_{yy} I_{yD} + I_{xy} I_{xD})\right) - \alpha \, \mathbf{Div} \, \psi_s' \left(|\nabla_3 u|^2 + |\nabla_3 v|^2\right) \nabla_3 v = 0 \quad (3.202)$$

Equations (3.201) and (3.202) are non-linear in argument $w = (u, v, 1)$. In order to solve the equations with common numerical methods, we need to express them as two linear set of equations. Fixed point iterations on w are used to deal with this non-linearity. A multi-scale (hierarchical) approach is combined with these fixed point iterations via a down-sampling strategy to better approximate the global optimum of the energy. k is used as an index to the pyramid level. The pyramid can go all the way to the top having the smallest possible sized images on the top level. Let $w^k = (u^k, v^k, 1)$, $k = 0, 1, ...$, with the initialization $w^0 = (0,0,1)$ at the coarsest grid (top level). Given a solution at w^k the solution for level $k+1$, denoted as w^{k+1} is found from:

$$\begin{aligned} &\psi_d' \left((I_D^{k+1})^2 + \gamma((I_{xD}^{k+1})^2 + (I_{yD}^{k+1})^2)\right)\left(I_x^k I_D^{k+1} + \gamma(I_{xx}^k I_{xD}^{k+1} + I_{xy}^k I_{yD}^{k+1})\right) \\ &\quad -\alpha \, \mathbf{Div} \, \psi_s' \left(|\nabla_3 u^{k+1}|^2 + |\nabla_3 v^{k+1}|^2\right) \nabla_3 u^{k+1} = 0 \end{aligned} \quad (3.203)$$

and

$$\begin{aligned} &\psi_d' \left((I_D^{k+1})^2 + \gamma((I_{xD}^{k+1})^2 + (I_{yD}^{k+1})^2)\right)\left(I_y^k I_D^{k+1} + \gamma(I_{yy}^k I_{yD}^{k+1} + I_{xy}^k I_{xD}^{k+1})\right) \\ &\quad -\alpha \, \mathbf{Div} \, \psi_s' \left(|\nabla_3 u^{k+1}|^2 + |\nabla_3 v^{k+1}|^2\right) \nabla_3 v^{k+1} = 0, \end{aligned} \quad (3.204)$$

Equations (3.203), (3.204) are still non-linear because of the non-linear term ψ' and the symbols I_*^{k+1}. In order to remove non-linearity in I_*^{k+1}, first-order Taylor expansions are used:

$$I_D^{k+1} \approx I_D^k + I_x^k du^k + I_y^k dv^k, \quad (3.205)$$

$$I_{xD}^{k+1} \approx I_{xD}^k + I_{xx}^k du^k + I_{xy}^k dv^k \text{ and} \quad (3.206)$$

$$I_{yD}^{k+1} \approx I_{yD}^k + I_{xy}^k du^k + I_{yy}^k dv^k. \quad (3.207)$$

$$(3.208)$$

Of course, $u^{k+1} = u^k + du^k$ and $v^{k+1} = v^k + dv^k$. As a result, we split the unknowns (u^{k+1}, v^{k+1}) into the solution in the previous step, (u^k, v^k) and unknown increments (du^k, dv^k).
For better readability let:

$$(\psi')_{Data}^k = \psi_d' \left((I_D^k + I_x^k du^k + I_y^k dv^k)^2 + \gamma((I_{xD}^k + I_{xx}^k du^k + I_{xy}^k dv^k)^2 + (I_{yD}^k + I_{xy}^k du^k + I_{yy}^k dv^k)^2)\right) \quad (3.209)$$

and

$$(\psi')_{Smooth}^k = \psi_s'(|\nabla_3(u^k + du^k)|^2 + |\nabla_3(v^k + dv^k)|^2). \quad (3.210)$$

Thus, Equations (3.203) and (3.204) can be written as:

$$\begin{aligned} &(\psi')_{Data}^k \left(I_x^k(I_D^k + I_x^k du^k + I_y^k dv^k) + \gamma\left[I_{xx}^k(I_{xD}^k + I_{xx}^k du^k + I_{xy}^k dv^k) + I_{xy}^k(I_{yD}^k + I_{xy}^k du^k + I_{yy}^k dv^k)\right]\right) \\ &\quad -\alpha \, \mathbf{Div}\left((\psi')_{Smooth}^k \nabla_3(u^k + du^k)\right) = 0 \end{aligned} \quad (3.211)$$

and

$$(\psi')_{Data}^k \left(I_y^k(I_D^k + I_x^k du^k + I_y^k dv^k) + \gamma\left[I_{yy}^k(I_{yD}^k + I_{xy}^k du^k + I_{yy}^k dv^k) + I_{xy}^k(I_{xD}^k + I_{xx}^k du^k + I_{xy}^k dv^k)\right]\right)$$

$$-\alpha\ \mathbf{Div}\left((\psi\,')^{k}_{Smooth}\nabla_3(v^k+dv^k)\right)=0 \tag{3.212}$$

This is still a nonlinear system of equations for a fixed k. Non-linearity is due to ψ'. We introduced another inner fixed point iteration loop in order to remove all non-linearity. Let there be a variable l which denotes the inner iteration in any level k. We initialize $du^{k,0} := 0$, $dv^{k,0} := 0$ and let $du^{k,l}, dv^{k,l}$ denote the iteration at some step l. Furthermore, let $(\psi\,')^{k,l}_{Data}$ and $(\psi\,')^{k,l}_{Smooth}$ be defined by Equations (3.209) and (3.210) above for some iteration k, l. Then, a linear system of equations in terms of $du^{k,l+1}$ and $dv^{k,l+1}$ is:

$$\begin{aligned}&(\psi\,')^{k,l}_{Data}\left(I^k_x(I^k_D+I^k_x du^{k,l+1}+I^k_y dv^{k,l+1})\right.\\&\left.+\gamma\left[I^k_{xx}(I^k_{xD}+I^k_{xx}du^{k,l+1}+I^k_{xy}dv^{k,l+1})+I^k_{xy}(I^k_{yD}+I^k_{xy}du^{k,l+1}+I^k_{yy}dv^{k,l+1})\right]\right)\\&-\alpha\ \mathbf{Div}\left((\psi\,')^{k}_{Smooth}\nabla_3(u^k+du^{k,l+1})\right)=0\end{aligned} \tag{3.213}$$

and

$$\begin{aligned}&(\psi\,')^{k,l}_{Data}\left(I^k_y(I^k_D+I^k_x du^{k,l+1}+I^k_y dv^{k,l+1})\right.\\&\left.+\gamma\left[I^k_{yy}(I^k_{yD}+I^k_{xy}du^{k,l+1}+I^k_{yy}dv^{k,l+1})+I^k_{xy}(I^k_{xD}+I^k_{xx}du^{k,l+1}+I^k_{xy}dv^{k,l+1})\right]\right)\\&-\alpha\ \mathbf{Div}\left((\psi\,')^{k,l}_{Smooth}\nabla_3(v^k+dv^{k,l+1})\right)=0.\end{aligned} \tag{3.214}$$

Hence, we fix $(\psi')^{k,l}_{Data}$,$(\psi')^{k,l}_{Smooth}$ and iterate on $du^{k,l+1}$,$dv^{k,l+1}$ to reach a solution. As soon as we reach a solution for $du^{k,l+1}$,$dv^{k,l+1}$ we update $(\psi')^{k,l}_{Data}$,$(\psi')^{k,l}_{Smooth}$ and start iterating on $du^{k,l+1}$,$dv^{k,l+1}$ for the new ψ' values. Eventually, we reach the optimal solution for that level and reach a fixed point solution for $\mathbf{w^k}$ This will then be used as the initial solution for $\mathbf{w^{k+1}}$ on the next finer level.

For better readability, let us express the equations into an easier format.

$$A(du^{k,l+1})+B(dv^{k,l+1})=E(du^{k,l+1}) \tag{3.215}$$
$$C(du^{k,l+1})+D(dv^{k,l+1})=F(dv^{k,l+1}) \tag{3.216}$$

where

$$A = (\psi')^{k,l+1}_{Data}\left(I^k_x I^k_x+\gamma(I^k_{xx}I^k_{xx}+I^k_{xy}I^k_{xy})\right), \tag{3.217}$$
$$B = (\psi')^{k,l+1}_{Data}\left(I^k_x I^k_y+\gamma(I^k_{xx}I^k_{xy}+I^k_{xy}I^k_{yy})\right), \tag{3.218}$$
$$C = (\psi')^{k,l+1}_{Data}\left(I^k_y I^k_x+\gamma(I^k_{yy}I^k_{xy}+I^k_{xy}I^k_{xx})\right), \tag{3.219}$$
$$D = (\psi')^{k,l+1}_{Data}\left(I^k_y I^k_y+\gamma(I^k_{yy}I^k_{yy}+I^k_{xy}I^k_{xy})\right), \tag{3.220}$$

$$E=\alpha\,(\psi\,')^{k,l}_{Smooth}\mathbf{Div}\left(\nabla_3(u^k+du^{k,l+1})\right)-(\psi\,')^{k,l}_{Data}\left(I^k_x I^k_D+\gamma(I^k_{xx}I^k_{xD}+I^k_{xy}I^k_{yD})\right) \tag{3.221}$$

and

$$F=\alpha\,(\psi\,')^{k,l}_{Smooth}\mathbf{Div}\left(\nabla_3(v^k+dv^{k,l+1})\right)-(\psi\,')^{k,l}_{Data}\left(I^k_y I^k_D+\gamma(I^k_{yy}I^k_{yD}+I^k_{xy}I^k_{xD})\right). \tag{3.222}$$

In equation (3.221) and (3.222), we need to compute $(u^k+du^{k,l+1})_{xx}$, $(u^k+du^{k,l+1})_{yy}$, $(v^k+dv^{k,l+1})_{xx}$ and $(v^k + dv^{k,l+1})_{yy}$. We can rearrange the terms as:

$$(u^k + du^{k,l+1})_{xx} + (u^k + du^{k,l+1})_{yy} = (u^k_{xx} + u^k_{yy}) + (du^{k,l+1}_{xx} + du^{k,l+1}_{yy}) \tag{3.223}$$

and

$$(v^k + dv^{k,l+1})_{xx} + (v^k + dv^{k,l+1})_{yy} = (v^k_{xx} + v^k_{yy}) + (dv^{k,l+1}_{xx} + dv^{k,l+1}_{yy}). \tag{3.224}$$

We use the approximation $X_{xx} + X_{yy} \approx \overline{X} - X$ for some scalar X, as suggested by Horn and Schunck [2] to express E and F as:

$$E = \alpha\,(\psi\,')^{k,l}_{Smooth}\left((\overline{u^k} - u^k) + (\overline{du^{k,l+1}} - du^{k,l+1})\right) - e \tag{3.225}$$

where,

$$e = (\psi\,')^{k,l}_{Data}\left(I^k_x I^k_D + \gamma(I^k_{xx} I^k_{xD} + I^k_{xy} I^k_{yD})\right) \tag{3.226}$$

and

$$F = \alpha\,(\psi\,')^{k,l}_{Smooth}\left((\overline{v^k} - v^k) + (\overline{dv^{k,l+1}} - dv^{k,l+1})\right) - f \tag{3.227}$$

where,

$$f = (\psi\,')^{k,l}_{Data}\left(I^k_y I^k_D + \gamma(I^k_{yy} I^k_{yD} + I^k_{xy} I^k_{xD})\right) \tag{3.228}$$

Now, our two equations are:

$$A(du^{k,l+1}) + B(dv^{k,l+1}) = \alpha\,(\psi\,')^{k,l}_{Smooth}\left(\overline{u^k} - u^k + \overline{du^{k,l+1}} - du^{k,l+1}\right) - e \tag{3.229}$$

$$C(du^{k,l+1}) + D(dv^{k,l+1}) = \alpha\,(\psi\,')^{k,l}_{Smooth}\left(\overline{v^k} - v^k + \overline{dv^{k,l+1}} - dv^{k,l+1}\right) - f \tag{3.230}$$

Grouping similar term together we obtain:

$$\left(A + \alpha\,(\psi\,')^{k,l}_{Smooth}\right)(du^{k,l+1}) + B(dv^{k,l+1}) = \alpha\,(\psi\,')^{k,l}_{Smooth}\left(\overline{u^k} + \overline{du^{k,l+1}} - u^k\right) - e \tag{3.231}$$

$$C(du^{k,l+1}) + \left(D + \alpha\,(\psi\,')^{k,l}_{Smooth}\right)(dv^{k,l+1}) = \alpha\,(\psi\,')^{k,l}_{Smooth}\left(\overline{v^k} + \overline{dv^{k,l+1}} - v^k\right) - f \tag{3.232}$$

which is expressed in matrix form as:

$$\begin{bmatrix} \left(A + \alpha(\psi')^{k,l}_{Smooth}\right) & B \\ C & \left(D + \alpha(\psi')^{k,l}_{Smooth}\right) \end{bmatrix} \begin{bmatrix} du^{k,l+1} \\ dv^{k,l+1} \end{bmatrix} = \begin{bmatrix} \alpha(\psi')^{k,l}_{Smooth}\left(\overline{u^k} + \overline{du^{k,l+1}} - u^k\right) - e \\ \alpha(\psi')^{k,l}_{Smooth}\left(\overline{v^k} + \overline{dv^{k,l+1}} - v^k\right) - f \end{bmatrix} \tag{3.233}$$

Figure 3.16 shows the computed 2D optical flow in NSERC datasets using the Brox method with intensity data only.

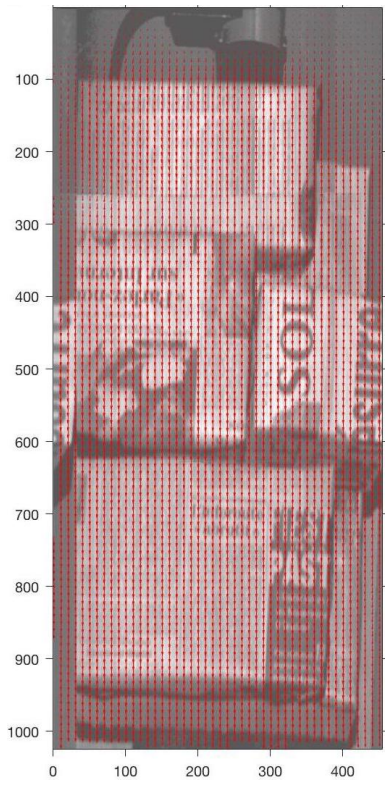


Figure 3.16: 2D optical flow using intensity data with Brox method

3.4.2 Range Flow by Brox et al. (Brox_r)

We extended 2D Brox et al.'s optical flow method to 3D range flow by using range motion constraint equation in our consideration. We are using the same Brox et al. [6] assumption constraints as follows:

Grayvalue constancy assumption: In range flow, we consider that the depth value does not change consecutive frames. This leads to range constraint equation:

$$Z_x u + Z_y v + w + Z_t = 0, \tag{3.234}$$

Gradient constancy assumption: The gradient of the image grayvalue does not vary due to displacement. However, we can use the gradient of the depth in range flow, which will give us:

$$\nabla Z(x, y, z, t) \quad = \quad \nabla Z(x + u, y + v, w, t + 1). \tag{3.235}$$

Smoothness assumption: We use the same smoothness as in the range flow regularization method.

$$E_{smooth} = \alpha^2(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2) \tag{3.236}$$

Now we have all our assumptions, we construct the 3D range energy functional. The data term is:

$$\begin{aligned} E_{Data}(u,v,w) &= \int_\Omega \psi\Big(|Z(x+u,y+v,z+w,t+1) - Z(x,y,z,t)|^2 \\ &+ \gamma\,|\nabla Z(x+u,y+v,z+w,t+1) - \nabla Z(x,y,z,t)|\Big)\,dxdydzdt. \end{aligned} \tag{3.237}$$

Here ψ does robust estimation, $\psi(s^2) = \sqrt{s^2+\epsilon}$. Brox et al. use $\epsilon = 0.001$.

A smoothness term that models the assumption of piecewise smoothness is:

$$E_{smooth}(u,v,w) = \int_\Omega \psi\left(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2\right) dxdydzdt. \tag{3.238}$$

Here $\nabla_4 = (\partial_x, \partial_y, \partial_z, \partial_t)^T$ is the spatio-temporal gradient. The total energy is the weighted sum between the data term and the smoothness term:

$$E(u,v,w) = E_{Data} + \alpha E_{smooth}, \tag{3.239}$$

with some regularization parameters $\alpha > 0$. Next, the goal is to find the functions u, v and w that minimize this energy.

According to the calculus of variations, a minimizer (3.239) must fulfill the Euler-Lagrange equations derived from the energy functional.

The Euler-Lagrange derivatives for this energy function can be defined using the following abbreviations:

$$Z_x = \partial_x Z(x+u,y+v,z+w,t+1), \tag{3.240}$$
$$Z_y = \partial_y Z(x+u,y+v,z+w,t+1), \tag{3.241}$$
$$Z_z = \partial_z Z(x+u,y+v,z+w,t+1), \tag{3.242}$$
$$Z_D = Z(x+u,y+v,z+w,t+1) - Z(x,y,z,t), \tag{3.243}$$
$$Z_{xx} = \partial_{xx} Z(x+u,y+v,z+w,t+1), \tag{3.244}$$
$$Z_{xy} = \partial_{xy} Z(x+u,y+v,z+w,t+1), \tag{3.245}$$
$$Z_{xz} = \partial_{xz} Z(x+u,y+v,z+w,t+1), \tag{3.246}$$
$$Z_{yy} = \partial_{yy} Z(x+u,y+v,z+w,t+1), \tag{3.247}$$
$$Z_{yz} = \partial_{yz} Z(x+u,y+v,z+w,t+1), \tag{3.248}$$
$$Z_{zz} = \partial_{zz} Z(x+u,y+v,z+w,t+1), \tag{3.249}$$
$$Z_{xD} = \partial_x Z(x+u,y+v,z+w,t+1) - \partial_x Z(x,y,z,t), \tag{3.250}$$
$$Z_{yD} = \partial_y Z(x+u,y+v,z+w,t+1) - \partial_y Z(x,y,z,t), \tag{3.251}$$
$$Z_{zD} = \partial_z Z(x+u,y+v,z+w,t+1) - \partial_z Z(x,y,z,t). \tag{3.252}$$

where F is the 3D range flow function to be minimized:

$$F = \psi_{Data} + \alpha\psi_{Smooth}. \tag{3.253}$$

Let $Z(x+w)$ denote $Z(x+u, y+v, z+w, t+1)$ and $Z(x)$ denote $Z(x, y, z, t)$. Then:

$$F = \psi\left(|Z(x+w) - Z(x)|^2 + \gamma(|\nabla Z(x+w) - \nabla Z(x)|^2\right) + \alpha\psi\left(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2\right). \quad (3.254)$$

Following the abbreviations in Equations (3.243), (3.250), (3.251) and (3.252), we have:

$$f = \psi\left(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2)\right) + \alpha\psi\left(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right). \quad (3.255)$$

$\psi(s^2)$ is defined as $\sqrt{s^2+\epsilon}$. Then the derivative of $\psi(s^2)$ is given as $\psi'(s^2) = \frac{1}{2\sqrt{s^2+\epsilon}}$

Required derivatives are:

$$\begin{aligned} f_u &= \psi'(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2)) \cdot \\ &\quad (Z_x Z_D + \gamma(Z_{xx} Z_{xD} + Z_{xy} Z_{yD} + Z_{xz} Z_{zD})), \end{aligned} \quad (3.256)$$

$$\begin{aligned} f_v &= \psi'(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2)) \cdot \\ &\quad (Z_y Z_D + \gamma(Z_{yx} Z_{xD} + Z_{yy} Z_{yD} + Z_{zy} Z_{zD})), \end{aligned} \quad (3.257)$$

$$\begin{aligned} f_w &= \psi'(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2)) \cdot \\ &\quad (Z_z Z_D + \gamma(Z_{zx} Z_{xD} + Z_{zy} Z_{yD} + Z_{zz} Z_{zD})). \end{aligned} \quad (3.258)$$

We can now derive the smoothness part as:

$$f_{u_x} = \psi'\left(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right) 2u_x, \quad (3.259)$$

$$f_{u_y} = \psi'\left(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right) 2u_y, \quad (3.260)$$

$$f_{u_z} = \psi'\left(uu_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right) 2u_z, \quad (3.261)$$

$$f_{u_t} = \psi'\left(uu_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right) 2u_t, \quad (3.262)$$

$$f_{v_x} = \psi'\left(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right) 2v_x, \quad (3.263)$$

$$f_{v_y} = \psi'\left(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right) 2v_y, \quad (3.264)$$

$$f_{v_z} = \psi'\left(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right) 2v_z, \quad (3.265)$$

$$f_{v_t} = \psi'\left(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right) 2v_t, \quad (3.266)$$

$$f_{w_x} = \psi'\left(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right) 2w_x, \quad (3.267)$$

$$f_{w_y} = \psi'\left(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right) 2w_y, \quad (3.268)$$

$$f_{w_z} = \psi'\left(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right) 2w_z. \quad (3.269)$$

$$f_{w_t} = \psi'\left(u_x^2 + u_y^2 + u_z^2 + u_t^2 + v_x^2 + v_y^2 + v_z^2 + v_t^2 + w_x^2 + w_y^2 + w_z^2 + w_t^2\right) 2w_t. \quad (3.270)$$

$$\frac{df_{u_x}}{dx} = \alpha u_{xx} \psi'_{Smooth},$$

$$
\begin{aligned}
\frac{df_{u_y}}{dy} &= \alpha u_{yy}\psi'_{Smooth}, \\
\frac{df_{u_z}}{dz} &= \alpha u_{zz}\psi'_{Smooth}. \\
\frac{df_{u_t}}{dt} &= \alpha u_{tt}\psi'_{Smooth}. \\
\frac{df_{v_x}}{dx} &= \alpha v_{xx}\psi'_{Smooth}, \\
\frac{df_{v_y}}{dy} &= \alpha v_{yy}\psi'_{Smooth}, \\
\frac{df_{v_z}}{dz} &= \alpha v_{zz}\psi'_{Smooth}, \\
\frac{df_{v_t}}{dt} &= \alpha v_{tt}\psi'_{Smooth}, \\
\frac{df_{w_x}}{dx} &= \alpha w_{xx}\psi'_{Smooth}, \\
\frac{df_{w_y}}{dy} &= \alpha w_{yy}\psi'_{Smooth}, \\
\frac{df_{w_z}}{dz} &= \alpha w_{zz}\psi'_{Smooth}. \\
\frac{df_{w_t}}{dt} &= \alpha w_{tt}\psi'_{Smooth}.
\end{aligned}
$$

The **Div** of $\nabla_4 u$ is $u_{xx} + u_{yy} + u_{zz} + +u_{tt}$. Similarly, the **Div** of $\nabla_4 v$ is $v_{xx} + v_{yy} + v_{zz} + v_{tt}$ and the **Div** of $\nabla_4 w$ is $w_{xx} + w_{yy} + w_{zz} + w_{tt}$. So:

$$\frac{df_{u_x}}{dx} + \frac{df_{u_y}}{dy} + \frac{df_{u_z}}{dz} + \frac{df_{u_t}}{dt} = \alpha \, \textbf{Div} \, \nabla_4 u \, \psi' \left(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2\right), \tag{3.271}$$

$$\frac{df_{v_x}}{dx} + \frac{df_{v_y}}{dy} + \frac{df_{v_z}}{dz} + \frac{df_{v_t}}{dt} = \alpha \, \textbf{Div} \, \nabla_4 v \, \psi' \left(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2\right), \tag{3.272}$$

$$\frac{df_{w_x}}{dx} + \frac{df_{w_y}}{dy} + \frac{df_{w_z}}{dz} + \frac{df_{w_t}}{dt} = \alpha \, \textbf{Div} \, \nabla_4 w \, \psi' \left(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2\right). \tag{3.273}$$

Therefore, the final Euler-Lagrange equations for 3D range flow using Brox et al.'s [6] method become:

$$\begin{aligned}
&\psi'(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2)) \cdot (Z_x Z_D + \gamma(Z_{xx} Z_{xD} + Z_{xy} Z_{yD} + Z_{xz} Z_{zD})) \\
&\qquad -\alpha \, \textbf{Div}(\psi'(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2)\nabla_4 u) = 0, \qquad (3.274) \\
&\psi'(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2)) \cdot (Z_y Z_D + \gamma(Z_{yx} Z_{xD} + Z_{yy} Z_{yD} + Z_{zy} Z_{zD})) \\
&\qquad -\alpha \, \textbf{Div}(\psi'(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2)\nabla_4 v) = 0, \qquad (3.275) \\
&\psi'(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2)) \cdot (Z_z Z_D + \gamma(Z_{zx} Z_{xD} + Z_{zy} Z_{yD} + Z_{zz} Z_{zD})) \\
&\qquad -\alpha \, \textbf{Div}(\psi'(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2)\nabla_4 w) = 0. \qquad (3.276)
\end{aligned}$$

Equations (3.274) to (3.276) are non-linear in argument $w = (u, v, w, 1)$. We employ fixed point iterations to deal with this non-linearity. A hierarchical approach is combined with these fixed point iterations using a downsampling strategy. k is used as an index to the pyramid level. Given a solution at level k as w^k, the solution for level $k+1$, w^{k+1} is found as:

$$\begin{aligned}
&\psi'((Z_D^{k+1})^2 + \gamma((Z_{xD}^{k+1})^2 + (Z_{yD}^{k+1})^2 + (Z_{zD}^{k+1})^2) \cdot \\
&\quad (Z_x^k Z_D^{k+1} + \gamma(Z_{xx}^k Z_{xD}^{k+1} + Z_{xy}^k Z_{yD}^{k+1} + Z_{xz}^k Z_{zD}^{k+1})) \\
&-\alpha\, \mathbf{Div}(\psi'(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2)\nabla_4 u) = 0,
\end{aligned} \tag{3.277}$$

$$\begin{aligned}
&\psi'((Z_D^{k+1})^2 + \gamma((Z_{xD}^{k+1})^2 + (Z_{yD}^{k+1})^2 + (Z_{zD}^{k+1})^2 \cdot \\
&\quad (Z_y^k Z_D^{k+1} + \gamma(Z_{yx}^k Z_{xD}^{k+1} + Z_{yy}^k Z_{yD}^{k+1} + Z_{zy}^k Z_{zD}^{k+1})) \\
&-\alpha\, \mathbf{Div}(\psi'(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2)\nabla_4 v) = 0,
\end{aligned} \tag{3.278}$$

$$\begin{aligned}
&\psi'((Z_D^{k+1})^2 + \gamma((Z_{xD}^{k+1})^2 + (Z_{yD}^{k+1})^2 + (Z_{zD}^{k+1})^2 \cdot \\
&\quad (Z_z^k Z_D^{k+1} + \gamma(Z_{zx}^k Z_{xD}^{k+1} + Z_{zy}^k Z_{yD}^{k+1} + Z_{zz}^k Z_{zD}^{k+1})) \\
&-\alpha\, \mathbf{Div}(\psi'(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2)\nabla_4 w) = 0.
\end{aligned} \tag{3.279}$$

This system of equations is still non-linear because of ψ' and variables of the form Z_*^{k+1}. To remove this non-linearity, we use 1^{st} order Taylor series expansions:

$$Z_D^{k+1} \approx Z_D^k + Z_x^k du^k + Z_y^k dv^k + Z_z^k dw^k, \tag{3.280}$$
$$Z_{xD}^{k+1} \approx Z_{xD}^k + Z_{xx}^k du^k + Z_{xy}^k dv^k + Z_{xz}^k dw^k, \tag{3.281}$$
$$Z_{yD}^{k+1} \approx Z_{yD}^k + Z_{yx}^k du^k + Z_{yy}^k dv^k + Z_{yz}^k dw^k, \tag{3.282}$$
$$Z_{zD}^{k+1} \approx Z_{zD}^k + Z_{zx}^k du^k + Z_{zy}^k dv^k + Z_{zz}^k dw^k, \tag{3.283}$$

where $u^{k+1} = u^k + du^k$, $v^{k+1} = v^k + dv^k$ and $w^{k+1} = w^k + dw^k$. Now the unknowns u^{k+1}, v^{k+1} and w^{k+1} are in terms of the known solution u^k, v^k and w^k and the unknowns du^k, dv^k and dw^k. We can write:

$$\begin{aligned}
(\psi')_{Data}^k &= \psi'((Z_D^k + Z_x^k du^k + Z_y^k dv^k + Z_z^k dw^k)^2 \\
&+ \gamma((Z_{xD}^k + Z_{xx}^k du^k + Z_{xy}^k dv^k + Z_{xz}^k dw^k)^2 \\
&+ (Z_{yD}^k + Z_{yx}^k du^k + Z_{yy}^k dv^k + Z_{yz}^k dw^k)^2 \\
&+ (Z_{zD}^k + Z_{zx}^k du^k + Z_{zy}^k dv^k + Z_{zz}^k dw^k)^2
\end{aligned} \tag{3.284}$$

and

$$(\psi')_{Smooth}^k = \psi'(|\nabla_4(u^k + du^k)|^2 + |\nabla_4(v^k + dv^k)|^2 + |\nabla_4(w^k + dw^k)|^2). \tag{3.285}$$

With these equations, Equations (3.277) to (3.279) can be rewritten as:

$$\begin{aligned}
0 &= (\psi')_{Data}^k \cdot (Z_x^k(Z_D^k + Z_x^k du^k + Z_y^k dv^k + Z_z^k dw^k)) \\
&+ \gamma(Z_{xx}^k(Z_{xD}^k + Z_{xx}^k du^k + Z_{xy}^k dv^k + Z_{xz}^k dw^k)
\end{aligned}$$

$$
\begin{aligned}
&+ \quad Z^k_{xy}(Z^k_{yD} + Z^k_{xy}du^k + Z^k_{yy}dv^k + Z^k_{yz}dw^k) \\
&+ \quad Z^k_{xz}(Z^k_{zD} + Z^k_{xz}du^k + Z^k_{zy}dv^k + Z^k_{zz}dw^k)) \\
&- \quad \alpha\ \mathbf{Div}((\psi')^k_{Smooth}\nabla_4(u^k + du^k)),
\end{aligned} \tag{3.286}
$$

$$
\begin{aligned}
0 \quad = \quad & (\psi')^k_{Data} \cdot (Z^k_y(Z^k_D + Z^k_x du^k + Z^k_y dv^k + Z^k_z dw^k)) \\
&+ \quad \gamma(Z^k_{yx}(Z^k_{xD} + Z^k_{xx}du^k + Z^k_{xy}dv^k + Z^k_{xz}dw^k) \\
&+ \quad Z^k_{yy}(Z^k_{yD} + Z^k_{xy}du^k + Z^k_{yy}dv^k + Z^k_{yz}dw^k) \\
&+ \quad Z^k_{yz}(Z^k_{zD} + Z^k_{xz}du^k + Z^k_{zy}dv^k + Z^k_{zz}dw^k)) \\
&- \quad \alpha\ \mathbf{Div}((\psi')^k_{Smooth}\nabla_4(v^k + dv^k)),
\end{aligned} \tag{3.287}
$$

$$
\begin{aligned}
0 \quad = \quad & (\psi')^k_{Data} \cdot (Z^k_z(Z^k_D + Z^k_x du^k + Z^k_y dv^k + Z^k_z dw^k)) \\
&+ \quad \gamma(Z^k_{zx}(Z^k_{xD} + Z^k_{xx}du^k + Z^k_{xy}dv^k + Z^k_{xz}dw^k) \\
&+ \quad Z^k_{zy}(Z^k_{yD} + Z^k_{xy}du^k + Z^k_{yy}dv^k + Z^k_{yz}dw^k) \\
&+ \quad Z^k_{zz}(Z^k_{zD} + Z^k_{xz}du^k + Z^k_{zy}dv^k + Z^k_{zz}dw^k)) \\
&- \quad \alpha\ \mathbf{Div}((\psi')^k_{Smooth}\nabla_4(w^k + dw^k)).
\end{aligned} \tag{3.288}
$$

This is still a non-linear system of equations because of ψ'. A second, inner, fixed point iteration is used to remove non-linearity in the increments. The non-linearity in ψ is resolved because ψ was chosen as a convex function and there is always a unique minimum solution. Let l denotes the inner iteration in level k. We set $du^{k,0} = 0$, $dv^{k,0} = 0$ and $dw^{k,0} = 0$ to initialize the increments and denote $du^{k,l}$, $dv^{k,l}$ and $dw^{k,l}$ as the increment values at inner iteration l. $(\psi')^{k,l}_{Data}$ and $(\psi')^{k,l}_{Smooth}$ are defined by Equations (3.284) and (3.285) for some k and l. We can obtain a linear system of equations in terms of $du^{k,l}$, $dv^{k,l}$ and $dw^{k,l}$:

$$
\begin{aligned}
0 \quad = \quad & (\psi')^{k,l}_{Data} \cdot (Z^k_x(Z^k_D + Z^k_x du^{k,l+1} + Z^k_y dv^{k,l+1} + Z^k_z dw^{k,l+1})) \\
&+ \quad \gamma(Z^k_{xx}(Z^k_{xD} + Z^k_{xx}du^{k,l+1} + Z^k_{xy}dv^{k,l+1} + Z^k_{xz}dw^{k,l+1}) \\
&+ \quad Z^k_{xy}(Z^k_{yD} + Z^k_{xy}du^{k,l+1} + Z^k_{yy}dv^{k,l+1} + Z^k_{yz}dw^{k,l+1}) \\
&+ \quad Z^k_{xz}(Z^k_{zD} + Z^k_{xz}du^{k,l+1} + Z^k_{zy}dv^{k,l+1} + Z^k_{zz}dw^{k,l+1})) \\
&- \quad \alpha\ \mathbf{Div}((\psi')^{k,l}_{Smooth}\nabla_4(u^k + du^{k,l+1})),
\end{aligned} \tag{3.289}
$$

$$
\begin{aligned}
0 \quad = \quad & (\psi')^{k,l}_{Data} \cdot (Z^k_y(Z^k_D + Z^k_x du^{k,l+1} + Z^k_y dv^{k,l+1} + Z^k_z dw^{k,l+1})) \\
&+ \quad \gamma(Z^k_{yx}(Z^k_{xD} + Z^k_{xx}du^{k,l+1} + Z^k_{xy}dv^{k,l+1} + Z^k_{xz}dw^{k,l+1}) \\
&+ \quad Z^k_{yy}(Z^k_{yD} + Z^k_{xy}du^{k,l+1} + Z^k_{yy}dv^{k,l+1} + Z^k_{yz}dw^{k,l+1}) \\
&+ \quad Z^k_{yz}(Z^k_{zD} + Z^k_{xz}du^{k,l+1} + Z^k_{zy}dv^{k,l+1} + Z^k_{zz}dw^{k,l+1})) \\
&- \quad \alpha\ \mathbf{Div}((\psi')^{k,l}_{Smooth}\nabla_4(v^k + dv^{k,l+1})),
\end{aligned} \tag{3.290}
$$

$$
0 \quad = \quad (\psi')^{k,l}_{Data} \cdot (Z^k_z(Z^k_D + Z^k_x du^{k,l+1} + Z^k_y dv^{k,l+1} + Z^k_z dw^{k,l+1}))
$$

$$
\begin{aligned}
&+ \quad \gamma(Z^k_{z,x}(Z^k_{xD} + Z^k_{xx}du^{k,l+1} + Z^k_{xy}dv^{k,l+1} + Z^k_{xz}dw^{k,l+1}) \\
&+ \quad Z^k_{zy}(Z^k_{yD} + Z^k_{xy}du^{k,l+1} + Z^k_{yy}dv^{k,l+1} + Z^k_{yz}dw^{k,l+1}) \\
&+ \quad Z^k_{zz}(Z^k_{zD} + Z^k_{xz}du^{k,l+1} + Z^k_{zy}dv^{k,l+1} + Z^k_{zz}dw^{k,l+1})) \\
&- \quad \alpha\ \mathbf{Div}((\psi')^{k,l}_{Smooth}\nabla_4(w^k + dw^{k,l+1})). \qquad (3.291)
\end{aligned}
$$

We can formulate our system in a nice and readable format. Suppose we have a system of three linear equations:

$$
\begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} =
\begin{bmatrix} J \\ K \\ L \end{bmatrix}.
$$

In matrix format, the solution of X, Y and Z is:

$$
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} =
\begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}^{-1}
\begin{bmatrix} J \\ K \\ L \end{bmatrix},
$$

For Equations (3.289), (3.290) and (3.291), we have three linear equation of the form:

$$
\begin{aligned}
A(du^{k,l+1}) + B(dv^{k,l+1}) + C(dw^{k,l+1}) &= J, \\
D(du^{k,l+1}) + E(dv^{k,l+1}) + F(dw^{k,l+1}) &= K, \\
G(du^{k,l+1}) + H(dv^{k,l+1}) + I(dw^{k,l+1}) &= L.
\end{aligned}
$$

Here, A through L can be written as:

$$
\begin{aligned}
A &= (\psi')^{k,l+1}_{Data}(Z^k_x Z^k_x + \gamma(Z^k_{xx}Z^k_{xx} + Z^k_{xy}Z^k_{xy} + Z^k_{xz}Z^k_{xz})), \\
B &= (\psi')^{k,l+1}_{Data}(Z^k_x Z^k_y + \gamma(Z^k_{xx}Z^k_{xy} + Z^k_{xy}Z^k_{yy} + Z^k_{xz}Z^k_{zy}), \\
C &= (\psi')^{k,l+1}_{Data}(Z^k_x Z^k_z + \gamma(Z^k_{xx}Z^k_{xz} + Z^k_{xy}Z^k_{yz} + Z^k_{xz}Z^k_{zz}), \\
D &= (\psi')^{k,l+1}_{Data}(Z^k_y Z^k_x + \gamma(Z^k_{yx}Z^k_{xx} + Z^k_{yy}Z^k_{xy} + Z^k_{yz}Z^k_{xz})), \\
E &= (\psi')^{k,l+1}_{Data}(Z^k_y Z^k_y + \gamma(Z^k_{yx}Z^k_{yx} + Z^k_{yy}Z^k_{yy} + Z^k_{yz}Z^k_{yz})), \\
F &= (\psi')^{k,l+1}_{Data}(Z^k_y Z^k_z + \gamma(Z^k_{yx}Z^k_{xz} + Z^k_{yy}Z^k_{yz} + Z^k_{yz}Z^k_{zz})), \\
G &= (\psi')^{k,l+1}_{Data}(Z^k_z Z^k_x + \gamma(Z^k_{zx}Z^k_{xx} + Z^k_{zy}Z^k_{xy} + Z^k_{zz}Z^k_{xz})), \\
H &= (\psi')^{k,l+1}_{Data}(Z^k_z Z^k_y + \gamma(Z^k_{zx}Z^k_{xy} + Z^k_{zy}Z^k_{yy} + Z^k_{zz}Z^k_{zy})), \\
I &= (\psi')^{k,l+1}_{Data}(Z^k_z Z^k_z + \gamma(Z^k_{zx}Z^k_{zx} + Z^k_{zy}Z^k_{zy} + Z^k_{zz}Z^k_{zz})), \\
J &= \alpha\ (\psi')^{k,l}_{Smooth}\mathbf{Div}(\nabla_4(u^k + du^{k,l+1})) - \\
&\quad (\psi')^{k,l}_{Data}(Z^k_x Z^k_D + \gamma(Z^k_{xx}Z^k_{xD} + Z^k_{xy}Z^k_{yD} + Z^k_{xz}Z^k_{zD})), \\
K &= \alpha\ (\psi')^{k,l}_{Smooth}\mathbf{Div}(\nabla_4(v^k + dv^{k,l+1})) - \\
&\quad (\psi')^{k,l}_{Data}(Z^k_y Z^k_D + \gamma(Z^k_{yx}Z^k_{xD} + Z^k_{yy}Z^k_{yD} + Z^k_{yz}Z^k_{zD})),
\end{aligned}
$$

$$
\begin{aligned}
L &= \alpha\,(\psi')^{k,l}_{Smooth}\mathbf{Div}(\nabla_4(w^k + dw^{k,l+1})) - \\
&\quad (\psi')^{k,l}_{Data}(Z^k_z Z^k_D + \gamma(Z^k_{zx}Z^k_{xD} + Z^k_{zy}Z^k_{yD} + Z^k_{zz}Z^k_{zD})).
\end{aligned}
$$

This linear system can be solved using iterative method SOR (Successive Over Relaxation).

Figure 3.17 shows the computed 3D range flow in NSERC datasets using the Brox method with range data only.

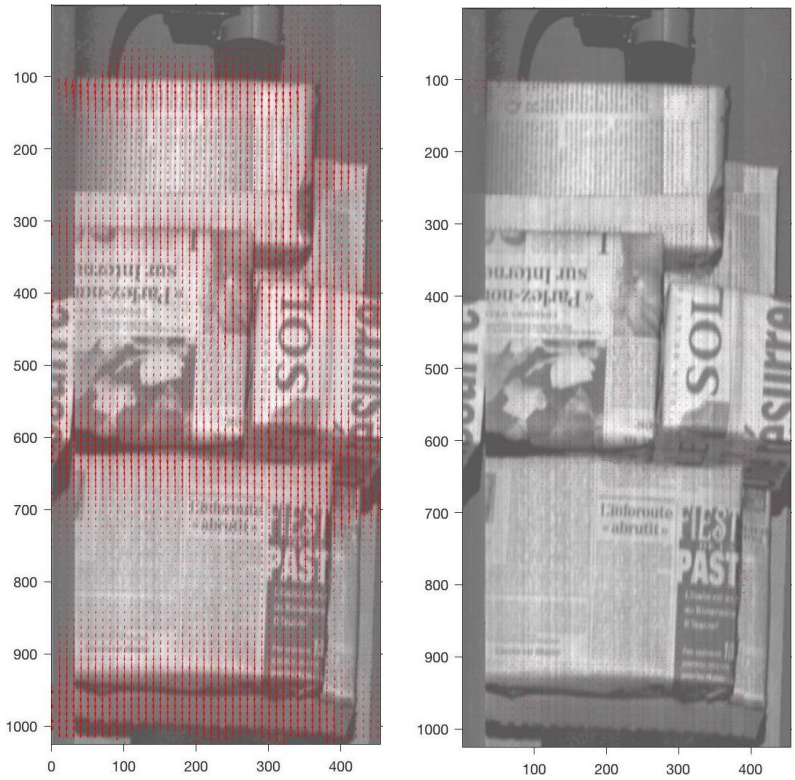


Figure 3.17: 3D range flow using range data with Brox's method (a) u,v components of 3D range flow, and (b) u,w components

3.4.3 Range Flow by Brox et al. Using Intensity and Range data (Brox_ir)

Range flow can be computed using Brox et al.'s approach that combines intensity and range data. Similar to the model in the previous section, Brox et al.'s constraints can be applied and combined for intensity and range data. We are using motion constraint equation and range motion constraint equation in our consideration. Therefore, 3D range flow can be computed by minimizing the following energy function.

$$E(u, v, w) = E_{Data} + \alpha E_{smooth}, \tag{3.292}$$

Data term becomes:

$$E_{Data}(u, v, w) = \int_\Omega \psi_d(|Z(x + u, y + v, z + w, t + 1) - Z(x, y, z, t)|^2$$

$$
\begin{aligned}
&+ \quad \gamma(|\nabla Z(x+u, y+v, z+w, t+1) - \nabla Z(x, y, z, t)|^2) \\
&+ \quad |I(x+u, y+v, t+1) - I(x, y, t)|^2 \\
&+ \quad \nabla(|I(x+u, y+v, t+1) - \nabla I(x, y, t)|^2) dxdydzdt. \qquad (3.293)
\end{aligned}
$$

While smoothness term:

$$
E_{smooth}(u, v, w) = \int_{\Omega} \psi_s \left(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2 \right) dxdydzdt. \qquad (3.294)
$$

Using same abbreviation from (3.180) to (3.187) and (3.240) to (3.252), the energy function can be written as:

$$
\begin{aligned}
f &= \psi_d \left(|Z(x+w) - Z(x)|^2 + \gamma(|\nabla Z(x+w) - \nabla Z(x)|^2 |I(x+w) - I(x)|^2 + \gamma(|\nabla I(x+w) - \nabla I(x)|^2 \right) \\
&+ \quad \alpha \psi_s \left(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2 \right). \qquad (3.295)
\end{aligned}
$$

Which is equivalent to:

$$
f = \psi_d \left(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2) \right) + \left(I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2) \right) + \alpha \psi_s \left(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2 \right). \qquad (3.296)
$$

According to the calculus of variations, a minimization of the energy function must fulfill the Euler-Lagrange equations derived from the energy functional.

$$
\begin{aligned}
f_u &= \psi'(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2) + I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2)) \cdot \\
&\quad (Z_x Z_D + \gamma(Z_{xx} Z_{xD} + Z_{xy} Z_{yD} + Z_{xz} Z_{zD}) + I_D I_x + \gamma(I_{xD} I_{xx} + I_{yD} I_{xy})), \qquad (3.297)
\end{aligned}
$$

$$
\begin{aligned}
f_v &= \psi'(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2) + I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2)) \cdot \\
&\quad (Z_y Z_D + \gamma(Z_{yx} Z_{xD} + Z_{yy} Z_{yD} + Z_{zy} Z_{zD}) + (I_D I_y + \gamma(I_{xD} I_{xy} + I_{yD} I_{yy}))), \qquad (3.298)
\end{aligned}
$$

$$
\begin{aligned}
f_w &= \psi'(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2) + I_z^2 + \gamma(I_{xz}^2 + I_{yz}^2)) \cdot \\
&\quad (Z_z Z_D + \gamma(Z_{zx} Z_{xD} + Z_{zy} Z_{yD} + Z_{zz} Z_{zD})). \qquad (3.299)
\end{aligned}
$$

The smoothness part can be derived from similar equations (3.259) to (3.270).

Therefore, the final Euler-Lagrange equations for 3D range flow using Brox et al.'s [6] method become:

$$
\begin{aligned}
&\psi'(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2) + I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2)) \cdot \\
&(Z_x Z_D + \gamma(Z_{xx} Z_{xD} + Z_{xy} Z_{yD} + Z_{xz} Z_{zD}) + I_D I_x + \gamma(I_{xD} I_{xx} + I_{yD} I_{xy})) \\
&-\alpha \, \mathbf{Div}(\psi'(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2) \nabla_4 u) = 0, \qquad (3.300)
\end{aligned}
$$

$$\begin{aligned}\psi'(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2) + I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2)) \cdot \\ (Z_y Z_D + \gamma(Z_{yx} Z_{xD} + Z_{yy} Z_{yD} + Z_{zy} Z_{zD}) + (I_D I_y + \gamma(I_{xD} I_{xy} + I_{yD} I_{yy}))) \\ -\alpha\ \mathbf{Div}(\psi'(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2)\nabla_4 v) = 0, \end{aligned} \tag{3.301}$$

$$\begin{aligned}\psi'(Z_D^2 + \gamma(Z_{xD}^2 + Z_{yD}^2 + Z_{zD}^2) + I_D^2 + \gamma(I_{xD}^2 + I_{yD}^2)) \cdot \\ (Z_z Z_D + \gamma(Z_{zx} Z_{xD} + Z_{zy} Z_{yD} + Z_{zz} Z_{zD})) \\ -\alpha\ \mathbf{Div}(\psi'(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2)\nabla_4 w) = 0. \end{aligned} \tag{3.302}$$

These equations are non-linear in argument $w = (u, v, w, 1)$. We employed fixed point iterations to deal with this non-linearity. A hierarchical approach was combined with these fixed point iterations using a downsampling strategy. k was used as an index to the pyramid level. Given a solution at level k as w^k, the solution for level $k+1$, w^{k+1} is found as:

$$\begin{aligned}\psi'((Z_D^{k+1})^2 + \gamma((Z_{xD}^{k+1})^2 + (Z_{yD}^{k+1})^2 + (Z_{zD}^{k+1})^2) + (I_D^{k+1})^2 + \gamma((I_{xD}^{k+1})^2 + (I_{yD}^{k+1})^2)) \cdot \\ (Z_x^k Z_D^{k+1} + \gamma(Z_{xx}^k Z_{xD}^{k+1} + Z_{xy}^k Z_{yD}^{k+1} + Z_{xz}^k Z_{zD}^{k+1}) + I_x^k I_D^{k+1} + \gamma(I_{xx}^k I_{xD}^{k+1} + I_{xy}^k I_{yD}^{k+1})) \\ -\alpha\ \mathbf{Div}(\psi'(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2)\nabla_4 u) = 0, \end{aligned} \tag{3.303}$$

$$\begin{aligned}\psi'((Z_D^{k+1})^2 + \gamma((Z_{xD}^{k+1})^2 + (Z_{yD}^{k+1})^2 + (Z_{zD}^{k+1})^2) + (I_D^{k+1})^2 + \gamma((I_{xD}^{k+1})^2 + (I_{yD}^{k+1})^2)) \cdot \\ (Z_y^k Z_D^{k+1} + \gamma(Z_{yx}^k Z_{xD}^{k+1} + Z_{yy}^k Z_{yD}^{k+1} + Z_{zy}^k Z_{zD}^{k+1} + I_y^k I_D^{k+1} + \gamma(I_{yy}^k I_{yD}^{k+1} + I_{xy}^k I_{xD}^{k+1})) \\ -\alpha\ \mathbf{Div}(\psi'(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2)\nabla_4 v) = 0, \end{aligned} \tag{3.304}$$

$$\begin{aligned}\psi'((Z_D^{k+1})^2 + \gamma((Z_{xD}^{k+1})^2 + (Z_{yD}^{k+1})^2 + (Z_{zD}^{k+1})^2 + (I_D^{k+1})^2 + \gamma((I_{xD}^{k+1})^2 + (I_{yD}^{k+1})^2)) \cdot \\ (Z_z^k Z_D^{k+1} + \gamma(Z_{zx}^k Z_{xD}^{k+1} + Z_{zy}^k Z_{yD}^{k+1} + Z_{zz}^k Z_{zD}^{k+1})) \\ -\alpha\ \mathbf{Div}(\psi'(|\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2)\nabla_4 w) = 0. \end{aligned} \tag{3.305}$$

This system of equations is non-linear because of ψ' and variables of the form Z_*^{k+1} and I_*^{k+1}. To remove this non-linearity, we use 1^{st} order Taylor series expansions as shown in Equations (3.205) to (3.207) and Equations (3.280) to (3.283).

where $u^{k+1} = u^k + du^k$, $v^{k+1} = v^k + dv^k$ and $w^{k+1} = w^k + dw^k$. Now, the unknowns u^{k+1}, v^{k+1} and w^{k+1} are in terms of the known solution u^k, v^k and w^k and the unknowns du^k, dv^k and dw^k. We can write:

$$\begin{aligned}(\psi')_{Data}^k &= \psi'((Z_D^k + Z_x^k du^k + Z_y^k dv^k + Z_z^k dw^k)^2 \\ &+ \gamma((Z_{xD}^k + Z_{xx}^k du^k + Z_{xy}^k dv^k + Z_{xz}^k dw^k)^2 \\ &+ (Z_{yD}^k + Z_{yx}^k du^k + Z_{yy}^k dv^k + Z_{yz}^k dw^k)^2 \\ &+ (Z_{zD}^k + Z_{zx}^k du^k + Z_{zy}^k dv^k + Z_{zz}^k dw^k)^2) \\ &+ (I_D^k + I_x^k du^k + I_y^k dv^k)^2 \\ &+ \gamma((I_{xD}^k + I_{xx}^k du^k + I_{xy}^k dv^k)^2 + (I_{yD}^k + I_{xy}^k du^k + I_{yy}^k dv^k)^2)) \end{aligned} \tag{3.306}$$

and

$$(\psi')^k_{Smooth} = \psi'(|\nabla_4(u^k + du^k)|^2 + |\nabla_4(v^k + dv^k)|^2 + |\nabla_4(w^k + dw^k)|^2). \tag{3.307}$$

With these equations, the three formulas can be rewritten as:

$$\begin{aligned}
0 &= (\psi')^k_{Data} \cdot (Z^k_x(Z^k_D + Z^k_x du^k + Z^k_y dv^k + Z^k_z dw^k)) \\
&+ \gamma(Z^k_{xx}(Z^k_{xD} + Z^k_{xx} du^k + Z^k_{xy} dv^k + Z^k_{xz} dw^k) \\
&+ Z^k_{xy}(Z^k_{yD} + Z^k_{xy} du^k + Z^k_{yy} dv^k + Z^k_{yz} dw^k) \\
&+ Z^k_{xz}(Z^k_{zD} + Z^k_{xz} du^k + Z^k_{zy} dv^k + Z^k_{zz} dw^k)) \\
&+ I^k_x(I^k_D + I^k_x du^k + I^k_y dv^k) \\
&+ \gamma(I^k_{xx}(I^k_{xD} + I^k_{xx} du^k + I^k_{xy} dv^k) \\
&+ I^k_{xy}(I^k_{yD} + I^k_{xy} du^k + I^k_{yy} dv^k)) \\
&- \alpha\, \mathbf{Div}((\psi')^k_{Smooth} \nabla_4(u^k + du^k)),
\end{aligned} \tag{3.308}$$

$$\begin{aligned}
0 &= (\psi')^k_{Data} \cdot (Z^k_y(Z^k_D + Z^k_x du^k + Z^k_y dv^k + Z^k_z dw^k)) \\
&+ \gamma(Z^k_{yx}(Z^k_{xD} + Z^k_{xx} du^k + Z^k_{xy} dv^k + Z^k_{xz} dw^k) \\
&+ Z^k_{yy}(Z^k_{yD} + Z^k_{xy} du^k + Z^k_{yy} dv^k + Z^k_{yz} dw^k) \\
&+ Z^k_{yz}(Z^k_{zD} + Z^k_{xz} du^k + Z^k_{zy} dv^k + Z^k_{zz} dw^k)) \\
&+ I^k_y(I^k_D + I^k_x du^k + I^k_y dv^k) \\
&+ \gamma(I^k_{yy}(I^k_{yD} + I^k_{xy} du^k + I^k_{yy} dv^k) \\
&+ I^k_{xy}(I^k_{xD} + I^k_{xx} du^k + I^k_{xy} dv^k)) \\
&- \alpha\, \mathbf{Div}((\psi')^k_{Smooth} \nabla_4(v^k + dv^k)),
\end{aligned} \tag{3.309}$$

$$\begin{aligned}
0 &= (\psi')^k_{Data} \cdot (Z^k_z(Z^k_D + Z^k_x du^k + Z^k_y dv^k + Z^k_z dw^k)) \\
&+ \gamma(Z^k_{zx}(Z^k_{xD} + Z^k_{xx} du^k + Z^k_{xy} dv^k + Z^k_{xz} dw^k) \\
&+ Z^k_{zy}(Z^k_{yD} + Z^k_{xy} du^k + Z^k_{yy} dv^k + Z^k_{yz} dw^k) \\
&+ Z^k_{zz}(Z^k_{zD} + Z^k_{xz} du^k + Z^k_{zy} dv^k + Z^k_{zz} dw^k)) \\
&- \alpha\, \mathbf{Div}((\psi')^k_{Smooth} \nabla_4(w^k + dw^k)).
\end{aligned} \tag{3.310}$$

This is a non-linear system of equations because of ψ'. A second, inner, fixed point iteration is used to remove the non-linearity in the increments. Let l denotes the inner iteration in level k. We set $du^{k,0} = 0$, $dv^{k,0} = 0$ and $dw^{k,0} = 0$ to initialize the increments and denote $du^{k,l}$, $dv^{k,l}$ and $dw^{k,l}$ as the increment values at inner iteration l. We can obtain a linear system of equations in terms of $du^{k,l}$, $dv^{k,l}$ and $dw^{k,l}$:

$$\begin{aligned}
0 &= (\psi')^{k,l}_{Data} \cdot (Z^k_x(Z^k_D + Z^k_x du^{k,l+1} + Z^k_y dv^{k,l+1} + Z^k_z dw^{k,l+1})) \\
&+ \gamma(Z^k_{xx}(Z^k_{xD} + Z^k_{xx} du^{k,l+1} + Z^k_{xy} dv^{k,l+1} + Z^k_{xz} dw^{k,l+1})
\end{aligned}$$

$$
\begin{aligned}
&+ \quad Z^k_{xy}(Z^k_{yD} + Z^k_{xy}du^{k,l+1} + Z^k_{yy}dv^{k,l+1} + Z^k_{yz}dw^{k,l+1}) \\
&+ \quad Z^k_{xz}(Z^k_{zD} + Z^k_{xz}du^{k,l+1} + Z^k_{zy}dv^{k,l+1} + Z^k_{zz}dw^{k,l+1})) \\
&+ \quad I^k_x(I^k_D + I^k_x du^{k,l+1} + I^k_y dv^{k,l+1}) \\
&+ \quad \gamma(I^k_{xx}(I^k_{xD} + I^k_{xx}du^{k,l+1} + I^k_{xy}dv^{k,l+1}) \\
&+ \quad I^k_{xy}(I^k_{yD} + I^k_{xy}du^{k,l+1} + I^k_{yy}dv^{k,l+1})) \\
&- \quad \alpha\, \mathbf{Div}((\psi')^{k,l}_{Smooth}\nabla_4(u^k + du^{k,l+1})),
\end{aligned} \tag{3.311}
$$

$$
\begin{aligned}
0 \quad &= \quad (\psi')^{k,l}_{Data} \cdot (Z^k_y(Z^k_D + Z^k_x du^{k,l+1} + Z^k_y dv^{k,l+1} + Z^k_z dw^{k,l+1})) \\
&+ \quad \gamma(Z^k_{yx}(Z^k_{xD} + Z^k_{xx}du^{k,l+1} + Z^k_{xy}dv^{k,l+1} + Z^k_{xz}dw^{k,l+1}) \\
&+ \quad Z^k_{yy}(Z^k_{yD} + Z^k_{xy}du^{k,l+1} + Z^k_{yy}dv^{k,l+1} + Z^k_{yz}dw^{k,l+1}) \\
&+ \quad Z^k_{yz}(Z^k_{zD} + Z^k_{xz}du^{k,l+1} + Z^k_{zy}dv^{k,l+1} + Z^k_{zz}dw^{k,l+1})) \\
&+ \quad I^k_y(I^k_D + I^k_x du^{k,l+1} + I^k_y dv^{k,l+1}) \\
&+ \quad \gamma(I^k_{yy}(I^k_{yD} + I^k_{xy}du^{k,l+1} + I^k_{yy}dv^{k,l+1}) \\
&+ \quad I^k_{xy}(I^k_{xD} + I^k_{xx}du^{k,l+1} + I^k_{xy}dv^{k,l+1})) \\
&- \quad \alpha\, \mathbf{Div}((\psi')^{k,l}_{Smooth}\nabla_4(v^k + dv^{k,l+1})),
\end{aligned} \tag{3.312}
$$

$$
\begin{aligned}
0 \quad &= \quad (\psi')^{k,l}_{Data} \cdot (Z^k_z(Z^k_D + Z^k_x du^{k,l+1} + Z^k_y dv^{k,l+1} + Z^k_z dw^{k,l+1})) \\
&+ \quad \gamma(Z^k_{zx}(Z^k_{xD} + Z^k_{xx}du^{k,l+1} + Z^k_{xy}dv^{k,l+1} + Z^k_{xz}dw^{k,l+1}) \\
&+ \quad Z^k_{zy}(Z^k_{yD} + Z^k_{xy}du^{k,l+1} + Z^k_{yy}dv^{k,l+1} + Z^k_{yz}dw^{k,l+1}) \\
&+ \quad Z^k_{zz}(Z^k_{zD} + Z^k_{xz}du^{k,l+1} + Z^k_{zy}dv^{k,l+1} + Z^k_{zz}dw^{k,l+1})) \\
&- \quad \alpha\, \mathbf{Div}((\psi')^{k,l}_{Smooth}\nabla_4(w^k + dw^{k,l+1})).
\end{aligned} \tag{3.313}
$$

Following same approach in the previous section, we can rewrite A,B,C,D,E,F,G,H,I,J,K and L. We now have three linear equations in the form:

$$
\begin{aligned}
A(du^{k,l+1}) + B(dv^{k,l+1}) + C(dw^{k,l+1}) &= J, \\
D(du^{k,l+1}) + E(dv^{k,l+1}) + F(dw^{k,l+1}) &= K, \\
G(du^{k,l+1}) + H(dv^{k,l+1}) + I(dw^{k,l+1}) &= L.
\end{aligned}
$$

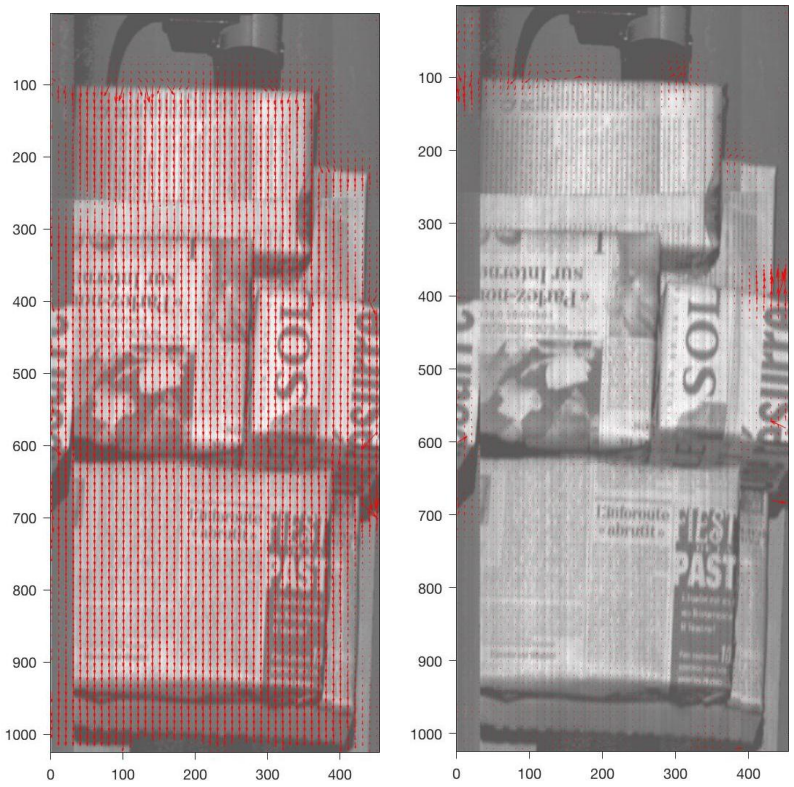
Here, A through L:

$$
\begin{aligned}
A &= (\psi')^{k,l+1}_{Data}((Z^k_x)^2 + \gamma((Z^k_{xx})^2 + (Z^k_{xy})^2 + (Z^k_{xz})^2) + (I^k_x)^2 + \gamma((I^k_{xx})^2 + (I^k_{xy})^2)), \\
B &= (\psi')^{k,l+1}_{Data}(Z^k_x Z^k_y + \gamma(Z^k_{xx}Z^k_{xy} + Z^k_{xy}Z^k_{yy} + Z^k_{xz}Z^k_{zy}) + I^k_x I^k_y + \gamma(I^k_{xx}I^k_{xy} + I^k_{xy}I^k_{yy})), \\
C &= (\psi')^{k,l+1}_{Data}(Z^k_x Z^k_z + \gamma(Z^k_{xx}Z^k_{xz} + Z^k_{xy}Z^k_{yz} + Z^k_{xz}Z^k_{zz})), \\
D &= (\psi')^{k,l+1}_{Data}(Z^k_y Z^k_x + \gamma(Z^k_{yx}Z^k_{xx} + Z^k_{yy}Z^k_{xy} + Z^k_{yz}Z^k_{xz}) + I^k_y I^k_x + \gamma(I^k_{yy}I^k_{xy} + I^k_{xy}I^k_{xx})), \\
E &= (\psi')^{k,l+1}_{Data}((Z^k_y)^2 + \gamma((Z^k_{yx})^2 + (Z^k_{yy})^2 + (Z^k_{yz})^2) + (I^k_y)^2 + \gamma((I^k_{yy})^2 + (I^k_{xy})^2)), \\
F &= (\psi')^{k,l+1}_{Data}(Z^k_y Z^k_z + \gamma(Z^k_{yx}Z^k_{xz} + Z^k_{yy}Z^k_{yz} + Z^k_{yz}Z^k_{zz})),
\end{aligned}
$$

$$
\begin{aligned}
G &= (\psi')^{k,l+1}_{Data}(Z^k_z Z^k_x + \gamma(Z^k_{zx} Z^k_{xx} + Z^k_{zy} Z^k_{xy} + Z^k_{zz} Z^k_{xz})),\\
H &= (\psi')^{k,l+1}_{Data}(Z^k_z Z^k_y + \gamma(Z^k_{zx} Z^k_{xy} + Z^k_{zy} Z^k_{yy} + Z^k_{zz} Z^k_{zy})),\\
I &= (\psi')^{k,l+1}_{Data}((Z^k_z)^2 + \gamma((Z^k_{zx})^2 + (Z^k_{zy})^2 + (Z^k_{zz})^2)),\\
J &= \alpha\,(\psi')^{k,l}_{Smooth}\mathbf{Div}(\nabla_4(u^k + du^{k,l+1})) -\\
&\quad (\psi')^{k,l}_{Data}(Z^k_x Z^k_D + \gamma(Z^k_{xx} Z^k_{xD} + Z^k_{xy} Z^k_{yD} + Z^k_{xz} Z^k_{zD}) + I^k_x I^k_D + \gamma(I^k_{xx} I^k_{xD} + I^k_{xy} I^k_{yD})),\\
K &= \alpha\,(\psi')^{k,l}_{Smooth}\mathbf{Div}(\nabla_4(v^k + dv^{k,l+1})) -\\
&\quad (\psi')^{k,l}_{Data}(Z^k_y Z^k_D + \gamma(Z^k_{yx} Z^k_{xD} + Z^k_{yy} Z^k_{yD} + Z^k_{yz} Z^k_{zD}) + I^k_y I^k_D + \gamma(I^k_{yy} I^k_{yD} + I^k_{xy} I^k_{xD})),\\
L &= \alpha\,(\psi')^{k,l}_{Smooth}\mathbf{Div}(\nabla_4(w^k + dw^{k,l+1})) -\\
&\quad (\psi')^{k,l}_{Data}(Z^k_z Z^k_D + \gamma(Z^k_{zx} Z^k_{xD} + Z^k_{zy} Z^k_{yD} + Z^k_{zz} Z^k_{zD})).
\end{aligned}
$$

This linear system can be solved using iterative method SOR (Successive Over Relaxation).

Figure 3.18 shows the computed 3D range flow in NSERC datasets using the Brox method when combining intensity and range data together.



(a) (b)

Figure 3.18: 3D range flow using intensity/range data with Brox method (a) u,v components of 3D range flow, and (b) u,w components

3.5 Conclusion

In this chapter, we explained several approaches to estimate 2D optical flow and 3D range flow in mathematical detail. Starting with well-known local approaches and global algorithms, then combining between local and global (CLG), lastly, Brox et al. approach. We showed the robust estimation formulas for each approach and displayed the NSERC dataset results for each method. In the next chapter, we describe the framework model design.

Chapter 4

Flow Model

In this chapter, we provide a detailed explanation of the techniques we used to construct the model of our framework. Starting from the derivative of the input images into the visualization of the output flow. We implement this framework in MATLAB_2018a. This builds on the previous chapter by demonstrating how we fit these approaches in our framework.

4.1 MATLAB Implementation

We implemented the framework using a programming language developed by MathWorks MATLAB_2018a in a Mac machine running the macOS High Sierra (10.13.6) operating system. MATLAB (Matrix Laboratory) is a tool that allows the user access to high performance numerical computation and visualization capabilities. With more than 50 sets of built-in functions available via toolboxes, MATLAB contains many collections of functions (toolboxes) written for specialized applications. For the work described herein, we used Image Processing Toolbox and Computer Vision Toolbox. In the next sub-sections, we describe the framework techniques that we implemented in MATLAB.

We implemented a model that can estimate 2D optical flow and 3D range flow of the motion in the scene. 2D Optical flow requires two intensity frames at time t and $t + 1$, while 3D range flow algorithms require depth frames at time t and $t + 1$ or RGB-D data (combined intensity and depth). Our 3D model uses a single view of the scene (not binocular) to estimate the flow in a variational method.

The diagram for the implemented framework shows in Figure 4.1.

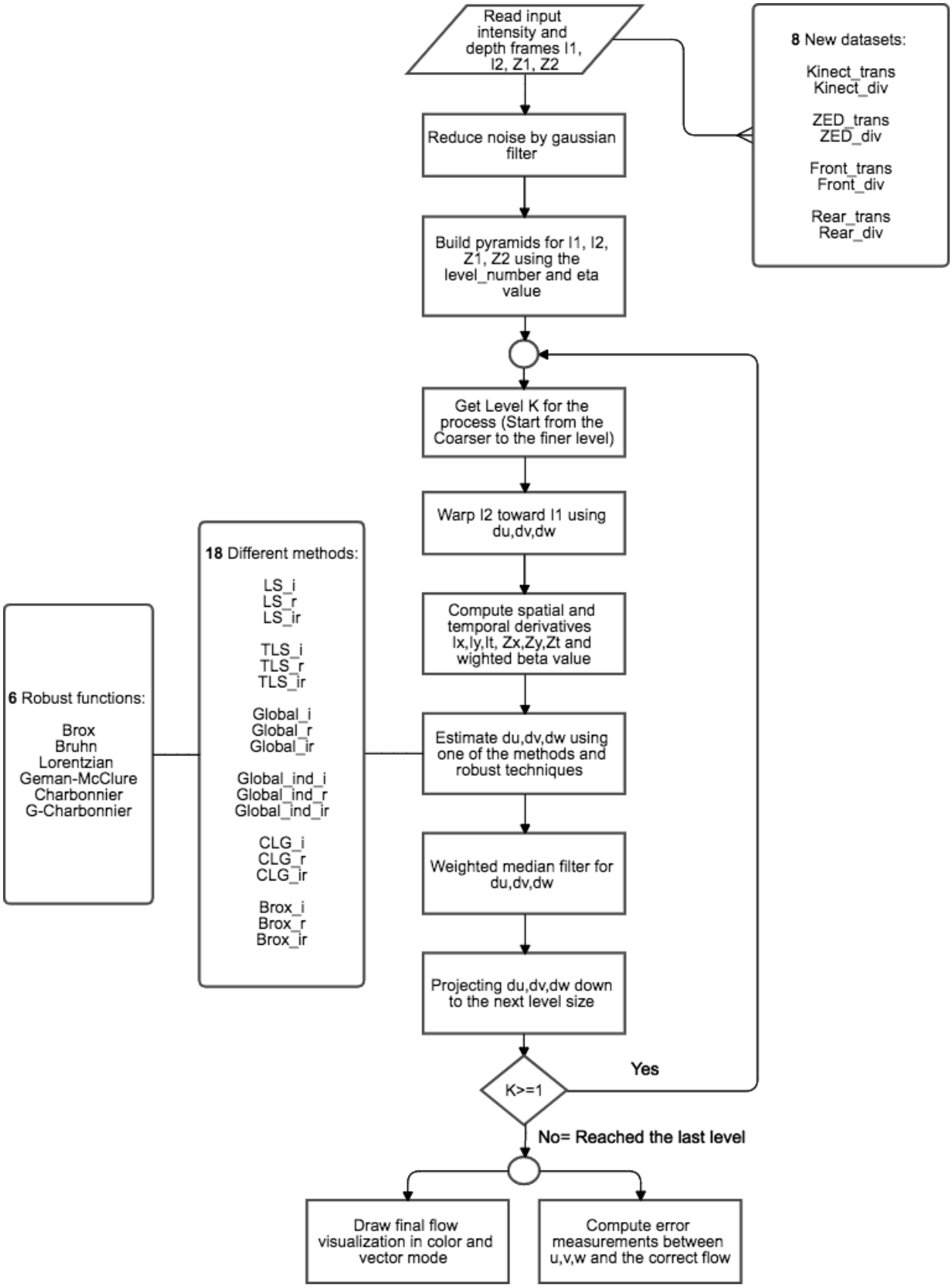


Figure 4.1: An overview of the implemented framework

4.1.1 Input Intensity and Depth Frames

Our framework required intensity and depth RGB-D frames as the input. We worked only with grayvalue images. Consequently, we converted the RGB to grayvalue images using *rgb2gray* function in MATLAB. In addition, as consistent with Spies et al.'s [3] framework, we read 4 intensity ($i1, i2, i3, i4$) and 4 depth ($z1, z2, z3, z4$) images. We then used box filter [0.25 0.5 0.25] that read the first three images ($i1, i2, i3$) and produced one intensity image $I1$. Finally, we sent the last three images ($i2, i3, i4$) to produce the second intensity $I2$. We applied the same process to depth images to produce $Z1$ and $Z2$. This step can improve the constant motion estimation between frames. If the motion between $i1$ and $i2$ and $i3$ and $i4$ are fixed, box filter enhances the ability to estimate the motion correctly.

4.1.2 Reduce Noise

Due to the high noise in the input images (either intensity or depth frames), we applied the *Gaussian* filter to reduce the influence of noise in further estimation steps. We used the gaussian filter with $\sigma = 0.5$. We avoided the use of high σ to preserve the edges and reduce over-smoothing.

4.1.3 Differentiation Using Filter

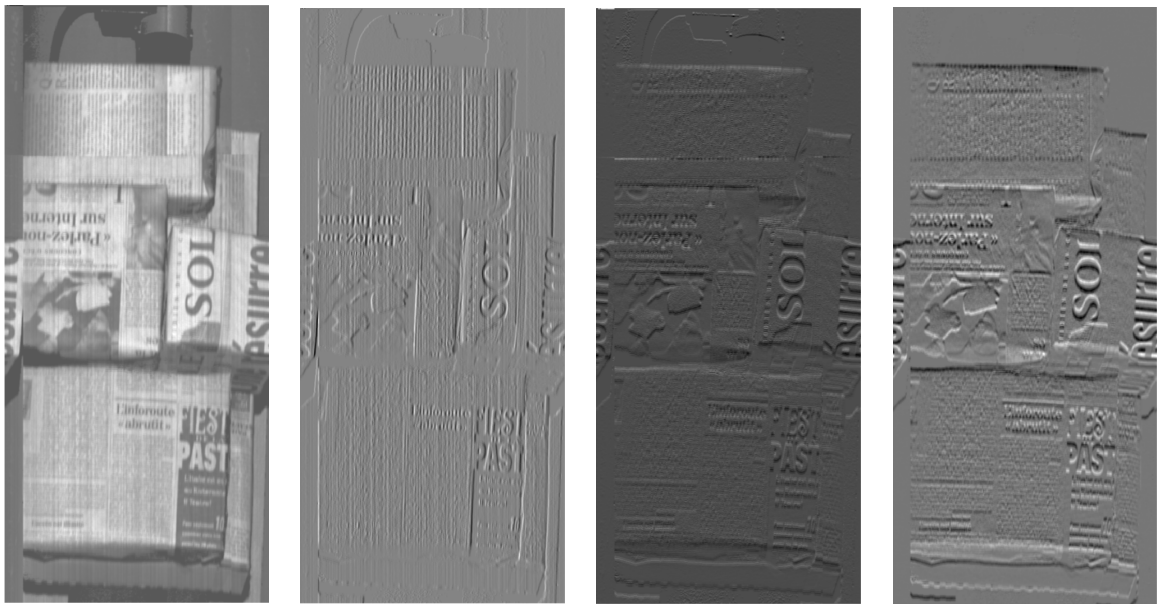
To compute 1st order derivatives of the intensity and depth images, we used Simoncelli's [175] 4 points central difference filtering. Thus, the 5 tap filter is $[-1\ 8\ 0\ -8\ 1]/12$. We used this filter because Spies et al. [3] used this differentiation filter in their original range flow method. In addition, the numerical calculations using 4 points central difference are better than those using 2 points central difference.

Brox et al.'s approaches (intensity, range, and intensity with range together) require 1st and 2nd order derivatives in the computation. We used filtering as suggested by Brox via email. For the 1st order derivatives, we used 4 points central difference filtering. The 5 tap filter for the is $[0.8666\ -0.6666\ 0.0\ 0.6666\ -0.8666]$. In the 2nd order derivatives, Brox suggested using the 3 tap filtering defined as: $[-0.5\ 0.0\ 0.5]$.

In our implementation for 2D optical and 3D range flow, we used 4 points central difference to compute the x and y spatial derivatives. Then, we used simple pixel differences to compute the temporal derivatives t. We applied this same process for both intensity and depth images. Figure 4.2 and Figure 4.3 show the derivatives for the NSERC datasets for intensity and depth, respectively.

4.1.4 Weighted the Intensity and Depth Differentiation

In the fusion intensity and depth approaches (which combine intensity and depth together in the same functional), we needed a weighting term β that could a count for the relative magnitude of the intensity and depth derivative. If the intensity derivatives are two orders of magnitude larger than depth derivatives. In that case, a value of $\beta = 0.5$ might be a better factor to weight the influence of the two types of derivatives.



I Ix Iy It

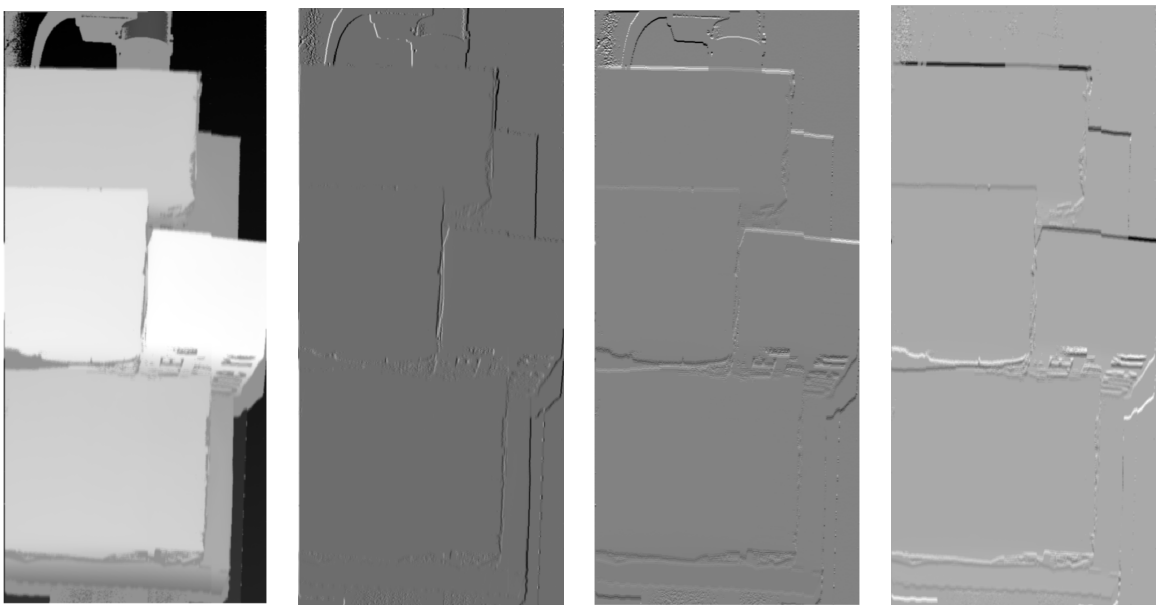
Figure 4.2: Intensity derivatives Ix,Iy and It for NSERC dataset

Alternatively, we could compute β as suggested by Spies et al. [97] as:

$$\beta^2 = \frac{< \|\nabla Z\|^2 >}{< \|\nabla I\|^2 >} \tag{4.1}$$

4.1.5 Hierarchical Technique

Faster motion, or motion in the edges, caused a large displacement that may be aliased (under-sampled in time). Generally, motion exeeding one pixel per frame is considered a large motion. To handle large motion, we used a hierarchical framework (coarse-to-fine model). We used the Brox et al. pyramid [6] because it is much more flexible than the standard Gaussian pyramid, which slows down image motion by a factor of 2 for each set of adjacent images in the pyramid. Given an image at one level in the Gaussian pyramid, the next higher image is generated by down-sampling the lower image in x and y by $\frac{1}{2}$ and then blurring that image by a Gaussian with the standard deviation $\sigma = 1.0$. The down-sampling amount in the Brox et al. pyramid is controlled by a variable (η) that does not have to be a power of 2. Indeed, η can be any values between $[0, 1]$. η having values of 0.8 or 0.95 are commonly used, while $\eta = 0.5$ gives the Gaussian pyramid as a special case in the Brox et al. pyramid. By using higher η values, some image artifacts are more easily maintained in the pyramid so that the averaging of 4 pixels into 1 pixel in the Gaussian pyramid is insufficient. In general, the Brox et al. pyramid is "taller" than a Gaussian pyramid in that it has more images with smaller changes in dimensions between adjacent images. We could adaptively determine the levels number in the pyramid using η value and the smaller width image so that we could start with the coarse level. For example, if we would like to start with width or height about 20, we might use the following formula to



Z Zx Zy Zt

Figure 4.3: Depth derivatives Zx,Zy and Zt for NSERC dataset

get the levels number:

$$Level\_number = round\left(\frac{log_{10}\left(\frac{20}{min(w,h)}\right)}{log_{10}(\eta)}\right) \tag{4.2}$$

Where w, h are the width and height of the input images and η can be chosen between $[0, 1]$. We are building equivalent pyramids for both intensity $I1, I2$ and depth $Z1, Z2$ frames.

4.1.6 Warping

Warping is a theoretically reasonable way to handle large displacements in a multi-resolution scheme. In each pyramid level, the 2nd image $I2$ warped using intermediate results (du, dv), or in the 3D case (du, dv, dw), toward the 1st image $I1$. Then, we computed the derivative of the warped image and performed the temporal averaging with the spatial derivative of the first image.

In essence, warping removes the current flow from the image. Thus, as processing continues down in the pyramid towards the original image, the flow between the 1st image and the warped 2nd image becomes smaller and smaller.

In 3D range flow, we compute (u, v, w) using depth data (Z) and its derivatives (Z_x, Z_y, and Z_t). In this case, the warping process need to remove (du, dv, dw) in each pyramid level. Therefore, we now have three computed flow components (du, dv, dw) that need to be added back into each pyramid level. (du, dv) can be removed similar to Brox et al.'s 2D pyramid method by performing 2D interpolation on images. Similarly, we can remove the dw component by re-interpolating $(dw, 1)$ for the warped image. In our framework, we can switch between bilinear or bicubic interpolation methods using $interp2$ function in MATLAB.

4.1.7 Weighted Median Filter

In our implementation, we applied the *weighted median filter* to intermediate flow during incremental estimation and warping to remove outliers between adjacent levels. The median filter makes the flow estimation more robust and improves the accuracy for the flow. In short, the median filter denoises the intermediate flow field. This idea was first introduced by Wedel et al. [176]. Sun et al. [177] then used the same observations about median filtering and L1 energy minimization from Li and Osher [178] to reformulate a non-local term and add modifcations to the data term to improve it. Each pixel is weighted by examining the structure of the surrounding pixels. The weights are determined by spatial distance, intensity distance, and occlusion state. To avoid over-smoothing and to preserve the edges and corner areas, we used $[5 \times 5]$ weighted median filter functions as provided by Sun et al. in [177].

4.1.8 Projecting Velocities

After estimating the flow at each level, the velocities should be projected down from a coarse level to a finer level. Since the pyramid levels have different widths and heights, we cannot simply copy the flow field to the next level. We have to distribute the flow in the next finer level (over-sample). Since the finer level is always bigger, we resized the velocities using the next-level size and the inverse of η that we used to build the pyramid. Figure 4.4 illustrates the coarse-to-fine framework.

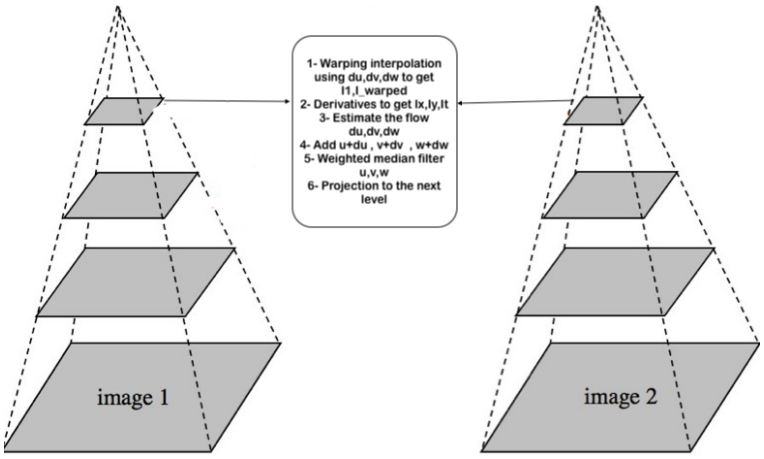


Figure 4.4: Coarse-to-fine framework: Left pyramid represent the first image while right pyramid represent the second image. The middle dialogue shows the intermediate process between two adjacent levels.

4.1.9 Robust Estimator

The main goal of robust statistics is to recover the structure that best fits the majority of the data while identifying and rejecting outliers. Robust estimation addresses the problem of finding the parameter values a that best fit a model $u(s;a)$ [8]. For example, in a local approach, the goal

is to find the model that best minimizes the size of the residual error using function ψ.

$$\psi\left((I_x u + I_y v + I_t)^2\right) \tag{4.3}$$

When the errors in the measurements are normally distributed, the optimal ψ function is quadratic. However, the problem with using the quadratic function is that the outliers are assigned a high weight. To increase the robustness of the flow, we should choose a ψ function that can reduce the influence of the outliers. There are several ψ functions that can be used, each with different motivation and strengths. To see this, Figure shows 4.5 different ψ functions and their influence. The influence function can be represented as a derivative of ψ' function.

In short, we applied robust techniques in our implementation to produce robust and dense optical flow, plus overcome the motion discontinuities problem [8] and to reduce the outliers [179]. The sections below discuss the different types of robust functions that we implemented.

Brox et al. and Bruhn et al. robust

Brox et al. [6] and Bruhn et al. [5],[4] used Charbonnier et al.'s [93] function differently as a robust function penalty. Brox et al. [6] used it as $\psi(s^2)$ which increases the concave and produces robust energy. This function can be applied to the data and smoothness terms separately. $\psi(s^2)$ defined as:

$$\psi(s^2) = \sqrt{s^2 + \epsilon^2} \tag{4.4}$$

The small positive constant ϵ keeps $\psi(s^2)$ convex, which helps the overall minimization process. Brox et al. chose $\epsilon = 0.0001$.

Then, the derivative of $\psi(s^2)$ with respect to s^2 is given as:

$$\psi'(s^2) = \frac{1}{2\sqrt{s^2 + \epsilon^2}} \tag{4.5}$$

Bruhn et al. [5, 4] used non-quadratic penalisers $\psi(s^2)$ defined as:

$$\psi(s^2) = 2\beta^2 \sqrt{1 + \frac{s^2}{\beta^2}} \tag{4.6}$$

where β is a scaling parameter. Then, the derivative of $\psi(s^2)$ with respect to s^2 is given as:

$$\psi'(s^2) = \frac{1}{\sqrt{1 + \frac{s^2}{\beta^2}}} \tag{4.7}$$

Black and Anandan robust

Black et al. in [8], [179] studied the Lorentzian and Geman-McClure functions as applied to local and global algorithms to estimate robust optical flow. The Lorentzian function is defined

as:

$$\psi(s^2) = log\left(1 + \frac{1}{2}\left(\frac{s}{\alpha}\right)^2\right) \tag{4.8}$$

Then, the derivative of $\psi(s^2)$ with respect to s^2 is given as:

$$\psi\,'(s^2) = \frac{2s}{2\alpha^2 + s^2} \tag{4.9}$$

While Geman-McClure function is defined as:

$$\psi(s^2) = \frac{s^2}{\alpha + s^2} \tag{4.10}$$

Then, the derivative of $\psi(s^2)$ with respect to s^2 is given as:

$$\psi\,'(s^2) = \frac{2s\alpha}{(\alpha + s^2)^2} \tag{4.11}$$

Sun et al. Studies

Sun et al. [177, 90] investigate several statistical robust functions and their influence on optical flow estimation algorithms. They tried Lorentzian and Charbonnier's functions in addition to the generalized Charbonnier penalty function $\psi(s^2) = \left(s^2 + \epsilon^2\right)^a$ that is equal to the Charbonnier penalty when $a = 0.5$ and non-convex when $a < 0.5$. They found a slightly non-convex penalty with $a = 0.45$, which performs consistently better than the Charbonnier penalty [90].

4.1.10 Energy Functional

Our framework applied a variational method to estimate 2D/3D flow. The use of a variational framework allowed us to properly handle discontinuities in the observed surfaces and in the 3-D displacement field to produce dense flow [140]. The optical/range flow algorithm estimates u, v, and w by minimizing an energy functional consisting of a data term (derived from constraints) and a smoothness term that enforces smooth and dense flow parameters:

$$E = \int_\Omega (E_D + E_S)dxdy, \tag{4.12}$$

where E_D and E_S are the data and smoothness term.

This energy functional needs to be minimized using a calculus variation of the Euler-Lagrange equation. As discussed in the previous chapter, we end up with a PDE (Partial Differential Equation) system that needs an iterative solution to converge. Our implementation uses the SOR method.

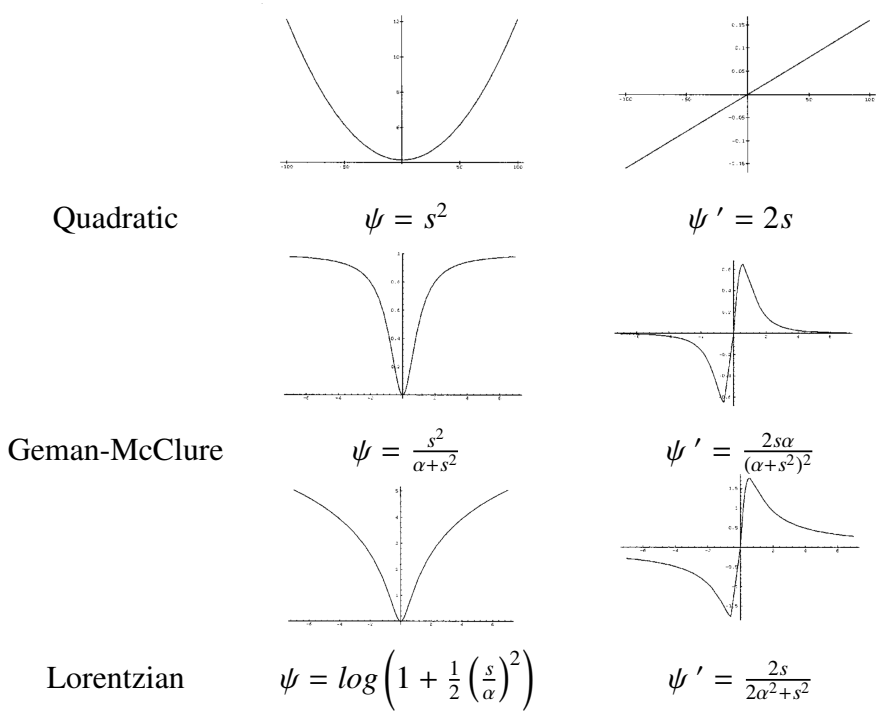


Figure 4.5: Some ψ and ψ' functions visualization [8]

Data term

To estimate 2D optical flow or 3D range flow, we must impose a set of assumptions in the motion field. As we previously stated, we used local, global, combined local/global (CLG) approaches, plus gradient constancy to model the motion field. The local approach assumes constant motion in a local neighbourhood. In this case, it is possible to locally solve the motion that best explains the local data and ignore the surrounding information using LS or TLS methods. On the other hand, a global approach assumes that the motion field is smooth in the surrounding neighbourhood. In the CLG approach, it is possible to model the field motion using the benefits from local and global approaches in the same formulation. Lastly, similar to the Brox et al. approach, we added the gradient constancy assumption for the surrounding neighbourhood to the motion field.

The data term in each approach depends on the constancy assumption shown in Table 4.1.

Smoothness term

In our global approaches, we used a popular homogeneous piecewise regularization term as defined in Spies et al. [3]. This smoothing method is used in common regularization approaches [6, 5]. The homogeneous diffusion is the simplest form of diffusion and describes a diffusion process that treats all directions equally.

For 2D optical flow, it is defined as:

$$E_S = |\nabla_3 u|^2 + |\nabla_3 v|^2 \tag{4.13}$$

While in 3D range flow the smoothness term looks like:

$$E_S = |\nabla_4 u|^2 + |\nabla_4 v|^2 + |\nabla_4 w|^2 \tag{4.14}$$

4.1.11 Removing Outliers

After estimation the flow for each pyramid level, we are removing the outliers of the intermediate flow. The percentage value is picked (3 works fine), then top (1-percentage) of velocities eliminated based on top large magnitudes being outliers. In the regularization approaches, we are eliminating the velocities that are five times larger the magnitude average.

4.1.12 Output Flow Visualization

Optical flow can be visualized by using Middlebury encoding as a colour scheme or as a vector field.

Colour Mode

In the colour mode, the optical flow can be encoded with an HSV colour space. To determine hue (H), we computed the angular between the optical flow vector and the x-axis in the image plane, so hue values have a range between $[0, 360°]$. The saturation (S) represents the ratio between the magnitude of each vector and the maximum magnitude among the whole motion field. Usually, the value (V) in the occluded area is set as 1 and the non-occluded area is set as 0. In this way, the motion vectors in the same direction share the same hue with different saturation to indicate the magnitude degree, the non-moving region is pure white, and the occluded region is black [180]. The colour coding scheme illustrated is demonstrated in Figure 4.6 (a). A further example of encoding 2D optical flow using colour mode is shown in Figure 4.6 (b).

(a) (b)

Figure 4.6: (a) Colour coding for optical flow field, (b) Example of encoding optical flow using colour coding using *ambush_5* frame in MPI Sintle datasets

Vectors Field Mode

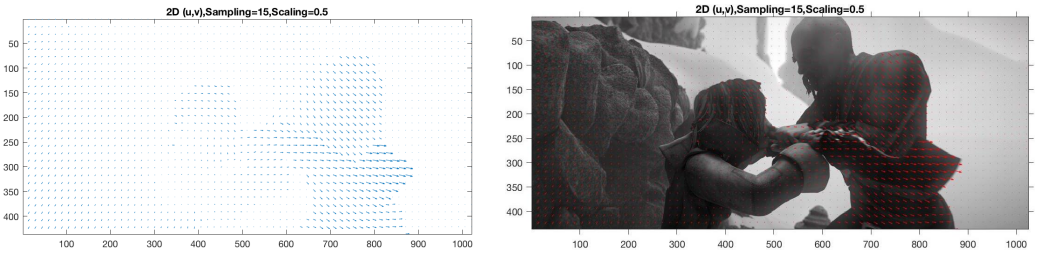
Optical flow can be visualized using a vector field. The arrow of the vectors represents the direction of the motion, while the size of the vectors represents the magnitude (the amount of

Table 4.1: Data terms and smoothness term in the implemented approaches

Method	**Data Term E_D**	**Constancy Assumption**	**Smoothness E_S**
LS_i	$I_x u + I_y v + I_t$	Intensity	$\times$
LS_r	$Z_x u + Z_y v + Z_z w + Z_t$	Range	$\times$
LS_ir	$Z_x u + Z_y v + Z_z w + Z_t +$ $\beta(I_x u + I_y v + I_t)$	Intensity + Range	$\times$
TLS_i	$I_x u + I_y v + I_t$	Intensity	$\times$
TLS_r	$Z_x u + Z_y v + Z_z w + Z_t$	Range	$\times$
TLS_ir	$Z_x u + Z_y v + Z_z w + Z_t +$ $\beta(I_x u + I_y v + I_t)$	Intensity + Range	$\times$
Global_i	$I_x u + I_y v + I_t$	Intensity	$\lvert\nabla_3 u\rvert^2 + \lvert\nabla_3 v\rvert^2$
Global_r	$Z_x u + Z_y v + Z_z w + Z_t$	Range	$\lvert\nabla_4 u\rvert^2 + \lvert\nabla_4 v\rvert^2 + \lvert\nabla_4 w\rvert^2$
Global_ir	$Z_x u + Z_y v + Z_z w + Z_t +$ $\beta(I_x u + I_y v + I_t)$	Intensity + Range	$\lvert\nabla_4 u\rvert^2 + \lvert\nabla_4 v\rvert^2 + \lvert\nabla_4 w\rvert^2$
Global_ind_i	$I_x u + I_y v + I_t$	Intensity in LS_i	$\lvert\nabla_3 u\rvert^2 + \lvert\nabla_3 v\rvert^2$
Global_ind_r	$Z_x u + Z_y v + Z_z w + Z_t$	Range in LS_r	$\lvert\nabla_4 u\rvert^2 + \lvert\nabla_4 v\rvert^2 + \lvert\nabla_4 w\rvert^2$
Global_ind_ir	$Z_x u + Z_y v + Z_z w + Z_t +$ $\beta(I_x u + I_y v + I_t)$	Intensity + Range in LS_ir	$\lvert\nabla_4 u\rvert^2 + \lvert\nabla_4 v\rvert^2 + \lvert\nabla_4 w\rvert^2$
CLG_i	$K_\rho * \left((I_x u + I_y v + I_t)^2\right)$	Intensity	$\lvert\nabla_3 u\rvert^2 + \lvert\nabla_3 v\rvert^2$
CLG_r	$K_\rho * \left((Z_x u + Z_y v + Z_z w + Z_t)^2\right)$	Range	$\lvert\nabla_4 u\rvert^2 + \lvert\nabla_4 v\rvert^2 + \lvert\nabla_4 w\rvert^2$
CLG_ir	$K_\rho * \left((Z_x u + Z_y v + Z_z w + Z_t)^2\right) +$ $\beta K_\rho * \left((I_x u + I_y v + I_t)^2\right)$	Intensity + Range	$\lvert\nabla_4 u\rvert^2 + \lvert\nabla_4 v\rvert^2 + \lvert\nabla_4 w\rvert^2$
Brox_i	$\lvert I(x+w) - I(x)\rvert^2 +$ $\gamma\lvert\nabla I(x+w) - \nabla I(x)\rvert^2$	Intensity + Intensity Gradient	$\lvert\nabla_3 u\rvert^2 + \lvert\nabla_3 v\rvert^2$
Brox_r	$\lvert Z(x+u, y+v, z+w, t+1) -$ $Z(x, y, z, t)\rvert^2 +$ $\gamma\lvert\nabla Z(x+u, y+v, z+w, t+1) -$ $\nabla Z(x, y, z, t)\rvert$	Range + Range Gradient	$\lvert\nabla_4 u\rvert^2 + \lvert\nabla_4 v\rvert^2 + \lvert\nabla_4 w\rvert^2$
Brox_ir	$(\lvert Z(x+u, y+v, z+w, t+1) -$ $Z(x, y, z, t) +$ $I(x+u, y+v, t+1) -$ $I(x, y, t)\rvert^2 +$ $\gamma\lvert\nabla Z(x+u, y+v, z+w, t+1) -$ $\nabla Z(x, y, z, t) +$ $\nabla I(x+u, y+v, t+1) -$ $\nabla I(x, y, t)\rvert^2$	Intensity + Intensity Gradient Range + Range Gradient	$\lvert\nabla_4 u\rvert^2 + \lvert\nabla_4 v\rvert^2 + \lvert\nabla_4 w\rvert^2$

the motion). For motion clarification, we sampled the areas to avoid overlapping the arrows. Using MATLAB, we can simply use the *quiver* function to show the vector field. In addition, we could show the imposition of the vector field on the intensity image. Figure 4.7 shows the vector field and imposed on the vectors of the intensity image for the same example in the

colour mode figure.



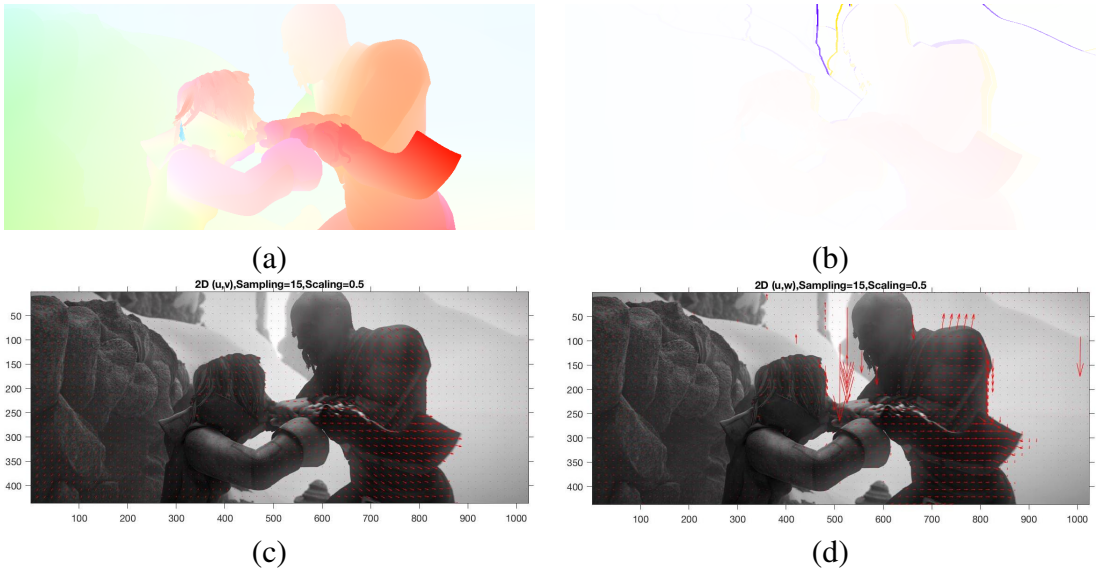
(a) (b)

Figure 4.7: (a) Vector field visualization (b) imposed vector field on the top of the intensity images (red just for representation) using *ambush_5* frame in MPI Sintle datasets

3D Plot

We estimated 3D range flow consisting of three components (u, v, w). In order to visualize the 3D flow, we could use the 2D plot in either colour mode or vector field method. We show (u, v) and (u, w) in separate plots. In addition, we might visualize the three components together in a vector field using *quiver3* MATLAB function. Figure 4.8 shows the colour and vector field visualization for the (u, v) and (u, w), respectively. Conversely, Figure 4.9 shows the 3D plot for the same motion.

Generally, we preferred vector field over colour scheme because colour representation often hides problems in the flow fields that are very visible in the vector fields.



(a) (b)

(c) (d)

Figure 4.8: (a) and (b) Colour representation for the 3D motion (u,v) and (u,w) respectively. Vector field visualization for (u,v) and (u,w) displayed in (c) and (d). Vector field imposed on the top of the intensity images using *ambush_5* frame in MPI Sintle datasets

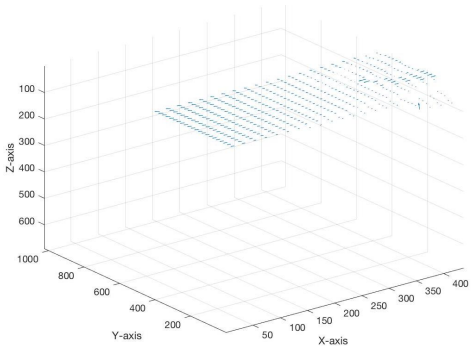


Figure 4.9: 3D Plot for (u, v, w) using cave frame in MPI Sintle datasets

4.1.13 Error Measurements

Using correct optical/range flow, we report 2D errors for optical flow and 3D errors for range flow using several error measurements. These error measurements can help provide quantitative and qualitative analysis of the range flow results. In the next section, we show how we measured 3D formulation errors and how we measured the same in 2D formulation using same concept, unless using 2D terms (u,v).

Average Angle Error (AAE)

We used Fleet [181] 3D average angular error to measure the error between correct 3D range flow and the estimated 3D range flow from the proposed algorithms. 3D velocity can be written as displacement per time unit as in $\tilde{\mathbf{v}} = (u, v, w)$ pixels/frame or as a space-time direction vector $(u, v, w, 1)$ in units of (pixel, pixel, pixel, frame). Let $\tilde{\mathbf{v}}_\mathbf{c} = (u_c, v_c, w_c, 1)$ represent the correct velocity at the pixel (i, j), and $\tilde{\mathbf{v}}_\mathbf{e} = (u_e, v_e, w_e, 1)$ represent the computed velocity at point (i, j, k). We can compute a normalized velocity using:

$$\hat{v} \equiv \frac{1}{\sqrt{u^2 + v^2 + w^2 + 1}}(u, v, w, 1)^T. \tag{4.15}$$

Let $\hat{\mathbf{v}}_\mathbf{c}$ and $\hat{\mathbf{v}}_\mathbf{e}$ be normalized versions of $\tilde{\mathbf{v}}_\mathbf{c}$ and $\tilde{\mathbf{v}}_\mathbf{e}$. The 3D angular error between the $\tilde{\mathbf{v}}_\mathbf{c}$ and $\tilde{\mathbf{v}}_\mathbf{e}$ is:

$$\text{AE} = \arccos(\hat{\mathbf{v}}_\mathbf{c} \cdot \hat{\mathbf{v}}_\mathbf{e}). \tag{4.16}$$

We compute the average of angle error and convert it from radian to degrees using the following formula:

$$\text{AAE} = \frac{1}{N}\sum \arccos\left(\frac{u_e u_c + v_e v_c + w_e w_c + 1}{\sqrt{(u_e^2 + v_e^2 + w_e^2 + 1)(u_c^2 + v_c^2 + w_c^2 + 1)}}\right). \tag{4.17}$$

Mean Absolute Error (MAE)

The mean absolute error is the mean of absolute error among all the pixels. The absolute error is AE computed as follows:

$$\text{AE} = |u_e - u_c| + |v_e - v_c| + |w_e - w_c| \tag{4.18}$$

So the MAE computed uses:

$$\text{MAE} = \frac{1}{N} \sum AE \tag{4.19}$$

where N number of pixels.

End Point Error (EPE)

The end point error is the average of the error among all the pixels. The error is E computed as follows:

$$\text{E} = |(\hat{v}_e - \hat{v}_c)| = \sqrt{(u_e - u_c)^2 + (v_e - v_c)^2 + (w_e - w_c)^2} \tag{4.20}$$

So the EPE computed uses:

$$\text{EPE} = \frac{1}{N} \sum E \tag{4.21}$$

Root Mean Squared Error (RMSE)

RMSE indicates both error distribution and overall accuracy level. It is computed using the following formula.

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum E^2} \tag{4.22}$$

Normalized Root Mean Square Error (NRMSE)

In addition, it is possible to measure the error in different scales by normalizing root mean squared error (NRMSE) using ground truth. This can be computed using the following formula:

$$\text{NRMSE} = \frac{RMS\,E}{max(\hat{v}_c) - min(\hat{v}_c)} \tag{4.23}$$

Pixel Percentage Errors

Pixels percentage errors can be computed using angle error (AE). We sorted the errors and then showed a percentage of pixels with angle error less than or equal to $(10°, 25°, 50°, 75°, 95°)$ degrees. In addition, can compute angle error in degrees of flow at $(10\%, 25\%, 50\%, 75\%, 95\%)$ percentiles. We are limiting the pixels analysis up to $95°$ because more than this degree mean the flow has wrong directions.

4.2 Conclusion

In this chapter, we discussed the design of our flow model. Starting from reading intensity and depth images into the visualization of the output flow. Also, we showed the derivative filter used in our implementation. In addition, we described our use of several energy functionals. Finally, we showed how we used three different robust estimator functions to produce dense flow. We also explained how error measurements (AAE, MAE, EPE, RMSE, and NRMSE) can be used to analyze the algorithms quantitatively and qualitatively. In the next chapter, we discuss range imaging sensors.

Chapter 5

Range Imaging and Sensors

Recently, there have been many advances in the computer vision field due to the improvement of computers and cameras. RGB-D cameras capture both colour images and depth information of the environment. Depth shows the distance between specific points in the scene; specifically, it is the distance between the camera/sensor and objects in the real world unit. Depth data greatly aids (3D) model construction, object tracking, motion detection, and so forth [182]. Range cameras/sensors can acquire depth information using different techniques. In our research, we generated a RGB-D dataset using several sensors in different times/positions. Furthermore, the inclusion of three space and time variables allow us to consider the 3D dataset as 4D. In this chapter, we describe several 3D imaging techniques in brief, before discussing the three different cameras/sensors that we used in our research. We used Kinect V2, ZED stereo camera, and iPhone X cameras to estimate 2D optical and 3D range flow. Several studies have tried to compute 2D optical flow and 3D scene flow using Kinect; however, while a few of these studies have tried to compute 3D range flow using Kinect V1, no studies have yet attempted Kinect V2.

5.1 Overview of 3D Imaging Techniques

Methods used to acquire range image information can be classified as either active or passive [42]. Passive methods rely only on radiation received from the environment, while active methods project a structure of coherent and/or modulated light on the target in order to obtain shape information [183]. There is a remarkable variety of 3D imaging techniques, and their classification is not unique [184]. Below, we briefly describe several techniques as mentioned in [184].

Stereo vision system: 3D information can be extracted by comparing the scene information from two or more image panels. Two or more cameras need to capture two differing views on a scene. Cameras need to be stereo rectified (have same focal length). In principle, extracting depth information requires the following sequence of processing steps: image acquisition, camera modelling (camera parameters focal length, central point, baseline), feature extraction, and triangulation (reconstruction) [184, 42].

Laser triangulation: Laser triangulation is a machine vision technique used to capture 3D

measurements by pairing a laser illumination source with a camera. Both single-point triangulation and laser stripes belong to this category. The laser beam and the camera are both aimed at the inspection target; however, by adopting a known angular offset between the laser source and the camera sensor, it is possible to measure depth differences using trigonometry [185].

Structured light: The structured light approach uses an active stereovision technique. A sequence of known patterns is sequentially projected onto a scene, which is deformed by geometric shapes of objects. The objects are then observed from a camera with a different viewpoint. By analyzing the distortion of the observed pattern and the disparity from the original projected pattern, we can extract the second viewpoint depth information [185, 42].

Time of flight: Range information can be extracted using the radar time of flight principle. The emitter unit generates a laser pulse, which impinges onto the target surface. A receiver detects the reflected pulse, and a suitable electronics measures the round trip travel time of the returning signal and its intensity [182].

Photogrammetry: A 3D model can be obtained using digital photographs. Photogrammetry is a science of making measurements from digital images. Camera calibration and orientation, image point measurements, 3D point cloud generation, surface generation, and texture mapping are all steps required to gain a 3D model of the scene using this technique [184].

Interferometry: In this method, the variation in surface geometry can be computed using the wavelength of the laser radiation. It operates by projecting a spatially or temporally varying periodic pattern onto a surface, followed by mixing the reflected light with a reference pattern. The reference pattern demodulate the signal to reveal the variation in surface geometry [184].

Shape from focusing: Depth map of the scene can be determined by using the focal properties of a lens. This technique can be passive or active. In a passive case, images are acquired with varying camera distance for the same scene. For each image and pixel, a focus measure is calculated in a local window. It also determines the spatial position of the image where this measure is maximal. Then, the depth map can be obtained by linking each pixel using the spatial positions for the pixels. The active version operates by projecting light onto the object to avoid difficulties in discerning surface texture [184, 186].

Shape from texture: This is a computer vision technique where a 3D object can be reconstructed from the 2D image. The idea is to find possible transformations of texture elements (texels) to reproduce the object surface orientation [184].

Shape from shadows: A 3D model of the unknown object can be reconstructed by capturing the shadow of a known object projected onto the target when the light is moving. This technique is different from structured light method [184].

Shape from shading: The shading technique gathers the shape of the target object by varying the position of the light source and acquiring the object from one viewing angle. The objects shape can be reconstructed by varying of the shading on the target surface [184].

Moir fringe range contours: This method constructed by lighting the scene with a light source through a grating and viewing the scene with a laterally displaced camera through a second identical grating. The resulting interference pattern shows equidistant contour lines [184].

5.2 Acquisition Range Data

As discussed in chapter two, previous scene/range flow studies have used several datasets for evaluating and testing their methods. These most common datasets are as follows: Middlebury dataset (*Teddy* and *Cones*) [91], Rotating Sphere datasets [140], EISATS dataset [187], KITTI datasets [32], MPI Sintel dataset [188], and, lastly, the Freiburg dataset [189]. These datasets generally provide stereo images sequences with ground truth disparity images. To use these available datasets in RGB-D range/scene flow methods, the disparity images d need to be converted to the depth map using focal length f and the baseline b of the camera based on this formula: $Z = (f * b)/d$.

None of these datasets can be considered as an idol dataset for RGB-D range/scene flow methods that need depth map as input. For example, KITTI datasets are suitable datasets for scene flow estimation methods that use binocular methods; however, they are not suitable for methods that need depth map because this method requires dense disparity ground truth for simulating depth. In sum, KITTI have missing values in both optical and disparity ground truth [180].

In order to construct RGB-D range data sequences with a ground truth range flow, we generated range sequences using different sensors. Then, we measured the correct optical and range flow for each motion and sequence. Generally, we tried to use popular consumer devices rather than expensive and complicated devices that require special labs and scientists to interpret.

To acquire range imaging, a scene that contains multiple 3D objects with a flat newspaper background was prepared in order to capture proper motion. We used newspapers to generate proper spatial derivatives. This is important because a solid colour background usually gives zero derivatives and optical flow differential techniques fail without these derivatives. In addition, we used several objects with different features, including corners, line and curve edges, to estimate the 2D/3D flow. In our scene, we organized a ball, a bucket, books, and other objects with curved edges to complicate the scene. Lastly, in Professor Barron's lab, we covered the windows by painting them a black colour to prevent sunlight motion from appearing in the scene.

The utilization cameras were situated approximately $1m$ from the scene on a unislider position that was connected to a controller NF90. The controller NF90 can move the unislider in horizontal and vertical directions in addition to rotation motion over a specified distance. To capture the sequences, cameras were moved vertically and horizontally (translation motions and divergence motions). Images were acquired every $1mm$. Twenty images were captured for a total motion $2cm$. Figure 5.1 shows the prepared scene and the unislider with the NF90 controller.

This research used three different camera, and three different techniques, to gather range information in the data acquisition process: the ZED camera uses stereo triangulation; the Kinect V2 uses Time-of-Flight (ToF); and the iPhone X has two cameras in the front and rear–the front camera uses a structured-light (SL) technique while the rear camera uses stereo triangulation technique. We describe the cameras in more detail below.

Figure 5.1: Prepared Scene to capture the sequences

5.3 Kinect V2

The Kinect sensor V1 was introduced in November 2010 by Microsoft for the Xbox 360 console system. The device contains an RGB camera, a depth sensor, an IR light source, a three-axis accelerometer, and a multi-array microphone, as well as supporting hardware that allows the unit to output sensor information to an external device. In February 2012, Microsoft released Kinect V2, which significantly improved the depth measurement accuracy. The new version of the Kinect sensor provides RGB, IR (Infrared), and depth images like its predecessor Kinect V1. However, the depth measurement mechanism and the image resolutions of the Kinect V2 differ from those of Kinect V1. For depth perception, the Kinect V1 uses the structured-light (SL) method, while the Kinect V2 uses the Time-of-Flight (ToF) method. Both range sensing principles, SL and ToF, are quite different and are subject to a variety of error sources [185]. Our research used Kinect V2 to acquire color, IR, and depth data. Kinect V1 and Kinect V2 can be seen in Figure 5.2. In addition, some comparative specifications of Kinect V1 and Kinect V2 are listed in Table 5.1.

(a) (b)

Figure 5.2: (a) Microsoft Kinect V1, and (b) Microsoft Kinect V2

The Kinect V2 provides an inexpensive option for acquiring 3D positional data in real-time (30 fps). With the Kinect, meaningful qualitative and quantitative data can be acquired and analyzed. The Kinect is an effective device with the potential to rapidly acquire data in many diverse situations [190]. RGB images, depth images, and IR images can all be acquired in real-time (30 fps).

Table 5.1: Comparative specifications of Kinect v1 and Kinect v2

	Kinect V1	**Kinect V2**
Resolution of color image	$640 \times 480(pixel)$	$1920 \times 1080(pixel)$
Resolution of IR and depth image	$320 \times 240(pixel)$	$512 \times 424(pixel)$
Field of view of color image	$62^\circ \times 48.6^\circ$	$84.1^\circ \times 53.8^\circ$
Field of view of IR and depth image	$58.5^\circ \times 46.6^\circ$	$70.6^\circ \times 60^\circ$
Maximum skeletal tracking	2	6
Method of depth measurement	Structured light	Time of Flight
Working range	$0.8 - 3.5m$	$0.5 - 8m$

5.3.1 Kinect Usage

Kinect V1 and V2 are used widely in computer vision studies even though it has been discontinued by Microsoft. The Kinect was originally invented to develop skeletal tracking, gesture recognition, and motion tracking in Xbox games [191]; consequently, it has had a significant impact beyond on the gaming industry, particularly in several computer vision applications. Several studies use Kinect in their applications. For example, [192, 193] used Kinect in physical rehabilitation tests and [194] used it for head and fall detection. In [195], Kinect helped develop an application encouraging older adults to maintain a physical exercise routine. In [196], Kinect imaging helped develop a lip-reading application for hearing-impaired people. Moreover, sign language recognition also uses Kinect [197]. It has also been used for dynamic characterization of soil micro-relief [198]. It could also be used for tracking objects such as in [199], which used it to track construction workers. In robot systems [200], gesture-based remote human-robot interaction is proposed using a Kinect in [201]. In dance, Kinect was used to develop an application to evaluate the dancer's performance [202]. Several studies have used Kinect to segment foreground and background or segment the objects as described in [49, 51, 203]. Other studies have used Kinect V1 data to estimate optical flow or range flow [111].

Our Usage: Using Windows 10 and the Kinect SDK 2.0, we used the tools in the SDK to capture colour and IR images with the maximum resolutions that Kinect V2 offers. Kinect SDK provides depth images as gray images (scaled between [0-255]). To get depth information, we used a special program that read the depth as 16-bit [204]. Figure 5.3 shows the Kinect V2 camera with the preparation scene. Some examples of colour, depth, and IR are shown in Figure 5.4. We applied some pre-processing steps on acquired data to prepare the range sequence, as explained in the next section.

5.3.2 Kinect pre-processing

Mirrored images

Since all data streams in Kinect V2 are mirrored, we reversed x-axis to get un-mirrored images. Therefore, all Kinect images (RGB, depth, and IR) were flipped using the MATLAB function $flip$ in the x-direction. Also, we converted colour images to grayscale format in order to use it in our algorithm.

Figure 5.3: Kinect V2 sensor and the scene.

Lens distortion

Depth and IR images have an apparent lens distortion problem. To correct lens distortion, we used MATLAB function *undistortImage*. This function requires intrinsic camera parameters that we collected out of the calibration process as explain in section 5.3.3.

Filling missing depth information

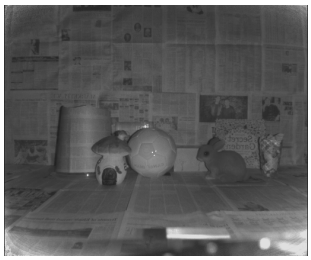
Due to the low quality and the noise in rendered depth frame images, depth images have regions with missing depth information–mostly in the edges. Kinect replaces this missing information with zero values. However, these regions need to be filled prior to using the depth frames in our algorithms. Zero value pixels were replaced by the statistical mod of the 25 surrounding pixels. This filling process returns sharper edges than statistical mean values because the mode calculation inserts the maximum occurring value in the surrounding 25 pixels to the center pixel [205].

Colour and depth alignment

The captured colour, IR, and depth images have different sizes and different fields of view, as shown in Table 5.1 and Figure 5.5. Our goal is to align RGB frames with the depth frame in the same size and field of view. Importantly, the aspect ratio of the original RGB is 0.56, but is 0.82 for depth/IR frames. According to [206], RGB and depth/IR frames can be cropped to remove extra parts and keep the mutual regions between colour and depth/IR cameras. Therefore, RGB frames are cropped from left and right, while depth frames are cropped from top and bottom. The new size for the depth frames is 512×360 with aspect ratio 0.7. RGB frames became 1080×1540 with aspect ratio 0.7. Thus, RGB frames can be resized to 512×360 without changing the aspect ratio.

Because RGB frames and depth frames have different channels, we need to register them to

(a)



(b)

(c)

Figure 5.4: Example of captured images by the Kinect V2 (Original images before pre-processing) (a) the colour image (b) the IR image (c) and the depth image (scale 0-255 for display)

get alignment frames. We registered RGB frames to IR frames using the MATLAB function *imregister*. IR frame was used because depth frame does not have enough features to register with RGB frame. Since the depth image is extracted from IR images, both have the same size and field of view and we can use the IR frame on behalf of depth frames. To show the output of registration, we used *imshowpair* MATLAB function for the registered and IR frames, as shown in Figure 5.6.

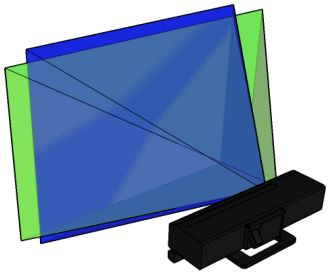


Figure 5.5: Kinect V2 RGB and IR cameras FOV.Blue represents the FOVs of the IR camera, while green irepresents the FOVs of the RGB camera.[9]

Figure 5.6: Registered RGB with IR frames

5.3.3 Kinect Calibration

Camera calibration estimates the cameras intrinsic, extrinsic, and lens-distortion parameters. These parameters can be used to correct for lens distortion, measure the size of an object in world units, and determine camera location in the scene. To calibrate Kinect V2, we utilized the Camera Calibration App in MATLAB from the Computer Vision System Toolbox. This app required taking several checkerboard images by the camera from different location and angles. We took twenty RGB and IR images for the checkerboard from different location and angles. We applied the same pre-processing steps as we did for our data sequences, such as flip, crop RGB and IR images, and then resizing the colour image to be the same size as the IR images. We then applied the calibration app for the colour and IR images separately. The parameters estimated during the calibration procedure are reported in Table 5.2 for color and IR images.

Table 5.2: Calibration parameters for RGB and IR camera in Kinect V2

	RGB	**IR**
Image size	512×360	512×360
Focal length	352.78	366.91
Principle points	[280.85 , 184.674]	[257.15 , 188.66]
Pixel size (μm)[9]	3.1	10
K1	0.0494	0.0819
K2	-0.0302	-0.2385
K3	-0.0565	0.0617
P1	0.0013	0.0031
P2	0.0015	-0.0007

5.4 ZED Camera

The ZED stereo camera was developed by Stereolabs [207], a privately owned, venture-backed company. The ZED camera is a passive stereo camera that attempts to reproduce images consistent with how human vision sees. Therefore, the camera uses its two cameras (left and right) to compute the space and motion of the 3D scene and triangulation to compute depth. Figure 5.7 shows the ZED Stereo camera [207]. With dual lenses, the ZED camera captures high-definition 3D video with a wide field of view. The video contains two synchronized left and

Figure 5.7: ZED stereo camera from Stereolabs

right video streams. This colour video is used by the ZED software on the host machine to create a depth map of the scene. We had access to the colour video and depth map simultaneously. In addition, the ZED camera uses a computer with NVIDIA GTX1060 to perform a real-time calculation (>30 fps).

- **Stereo Capture:** There are several video modes available with different output resolutions and frame rates, as shown in Table 5.3. In addition, the output images can be in different formats including rectified, unrectified, and grayscale images with view selection (left, right, or side-by-side).
- **Depth Reception:** Depth perception is the ability to determine the distance between objects and perceive scenes in three dimension. The ZED camera has two cameras separated by a baseline of 12*cm*, which capture a high-resolution 3D video of the scene and estimate depth and motion by comparing the displacement of pixels between the left and right images. The ZED camera can perceive depth between 50*cm* (1.8 feet) and 20 meters (65 feet). The frame rate of depth capture can be as high as 100 FPS. The field of view is large, up to 110°. The ZED stores a distance value (Z) for each pixel (X, Y) in the image. The distance is expressed in metric units and calculated from the back of the left camera to the scene object.

Table 5.3: Video Modes available in ZED Camera

Video Mode	**Output Resolution (side by side)**	**Frame Rate (fps)**	**Field of View**
2.2K	4416×1242	15	Wide
1080p	3840×1080	30, 15	Wide
720p	2560×720	60, 30, 15	Extra Wide
WVGA	1344×376	100, 60, 30, 15	Extra Wide

5.4.1 ZED Usage

Recently, several studies have used ZED stereo camera in their applications. The ZED camera can be applied in autonomous robot navigation, virtual reality, tracking, motion analysis, and so forth. Other studies use ZED images to test their proposed algorithms, such as in [208]'s evaluation of the deblurring algorithm. In [209], they estimated the disparity map using optical

flow in ZED images. In [210], they developed an indoor mapping system for smart city applications using Kinect and ZED cameras. Similarly, [211] has proposed using the ZED stereo camera in injury evalution applications. Currently, the most novel application is in Advanced Driver Assistance Systems (ADAS) and other automotive systems. ZED stereo cameras play a significant role in automotive systems-related applications. In 2018, Varma et al. [212] developed a stereo system that can detect and inform the driver about upcoming unmarked and marked speed bumps and give the vehicle's distance from the object in real-time using deep learning techniques. Rahul et al. [213] have also proposed a system to detect and tag upcoming objects and estimate its distance. Similarly, [214] also propose a automotive system that can detect obstacles in stereo vision. Despite these innovative approaches, no study has yet estimated 2D optical and 3D range flow using ZED camera images.

Our usage: Using Windows 10 and the ZED SDK, we used the provided ZED tools to capture left, right, and depth images with 1080p resolution. We captured colour and depth images using ZED camera in the same size. Colour images were converted to grayscale for use in our range flow algorithm. However, we first had to fill in the missing information in the depth images as we did for Kinect depth images. Figure 5.8 shows the prepared scene using the ZED camera. Some example of colour, depth, and IR are shown in Figure 5.9.

Figure 5.8: ZED Camera with captured scene

5.4.2 ZED Calibration

Stereolabs provides tools to help achieve the calibration parameters by following the specific steps in their SDK. The next table lists 5.4 the ZED camera parameters for left images in 1080p as used in our research.

(a) (b)

(c)

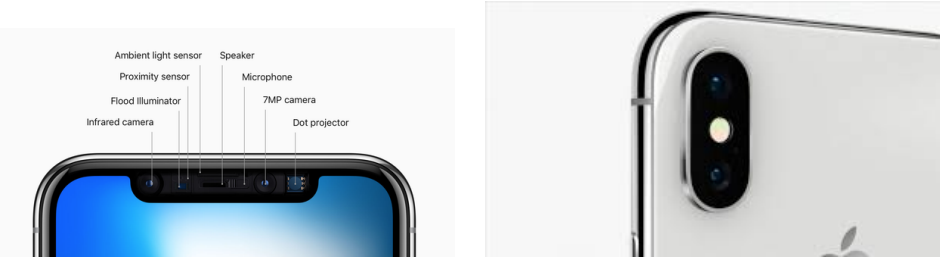
Figure 5.9: Example of captured images by the ZED camera (a) The left image (b) The right image (c) The depth image (scale 0-255 for display)

5.5 iPhone X

In September 2017, Apple Inc announced their next smartphone named the iPhone X [215]. Newer updates to the iPhone X are iPhone XS and iPhone XR, which have similar front and rear cameras. The rear camera is a dual (two camera) 12 megapixel wide-angle camera. This iPhone model also has a specific photo mode, Portrait Mode, that can capture photos while hiding depth information on them. The iPhone X calculates depth information using the stereo triangulation technique [215]. The front camera is a TrueDepth camera. TrueDepth starts with a traditional 7 megapixel front-facing selfie camera. It adds an infrared emitter that projects over 30,000 dots in a known pattern onto the users face. Those dots are then photographed by a dedicated infrared camera for analysis. There is also a proximity sensor, presumably so that the system knows when a user is close enough to activate. An ambient light sensor helps the system set output light levels. This camera provides depth information that is extracted by a structured light (SL) technique. Its depth estimation works by having an IR emitter send out 30,000 dots arranged in a regular pattern. They are visible to the IR camera that reads the deformed pattern as it reflects off surfaces at various depths. Figure 5.10 shows the front and the rear cameras in iPhone X.

Table 5.4: Calibration parameters for ZED camera in 1080P mode

	Left Camera	**Right Camera**
Image size	1920 ×1080	1920 ×1080
Focal length	1397.9	1399.07
Principle points	[997.1 , 571.6]	[975.158 , 622.125]
Pixel size (μm)	2	10
K1	0.0268908	-0.17122
K2	-0.170613	0.0266326



(a) (b)

Figure 5.10: Front and rear cameras in the iPhone X

5.5.1 iPhone X Usage

iPhone images are used in several studies. For example, in [216], the author constructed 3D object measurements using iPhone X and Kinect images. In [217], they described an application for iPhone X that monitors blood pressure. Previous studies have considered images from earlier iPhone models. Applications using optical flow for the iPhone 6 have been considered by [218, 219, 220]. [221] designed an application for the iPhone 4S using optical flow and structure from motion algorithms to capture a page spread of a book by sweeping the phone across the open book. However, no study has yet estimated 2D and 3D range flow using iPhone X front or rear cameras images.

Our usage: The iPhone X was situated in a landscape position in front of a prepared scene. We then captured the required motion using a small tripod for the iPhone X. We also used a remote control connected via Bluetooth to the iPhone X to capture the images without physically touching the iPhone X. While capturing the sequences, we adjusted the camera settings to Portrait Mode for both the front and rear cameras. Figure 5.11 shows the prepared scene with iPhone X. To extract depth information from the iPhone X images, we wrote a SWIFT code (Swift is a powerful and intuitive programming language for macOS, iOS, watchOS, tvOS, and beyond). Depth information was stored inside the image object as auxiliary data. Our Swift code used the portrait images as input, extracted the depth information, and then subsequently saved it separately as a png file to be used in our tested algorithms. Notably, the extracted depth images are smaller than color images. Once we adjsuted color and depth images to have the same aspect ratio of 0:75, we re-scaled the color image size to the same size as the depth images as advised in WWDC2017 [79]. Also, depth images were flipped and rotated to align with the color images. Figure 5.12 shows an example of the captured images using the

Truedepth camera (front) and extracted depth image, while Figure 5.13 shows images using stereo cameras (rear) and extracted depth images.

Figure 5.11: iPhone X with a captured scene

(a) (b)

Figure 5.12: Colour and extracted depth images using the front camera in iPhone X

5.5.2 iPhone X Calibration

For iPhone X, we took twenty images for the checkerboard using front and rear cameras in Portrait Mode. We resized the RGB images to the same size as depth images. Then we used the Camera Calibration App in MATLAB from the Computer Vision System Toolbox. The table 5.5 bellows lists the camera parameters for the front and rear cameras.

5.6 Orbbec Persee

The Orbbec company was founded in 2013 by a group of passionate engineers and researchers [10]. It provides several 3D imaging products including Persee, Astra Series, and Astra Stereo.

(a) (b)

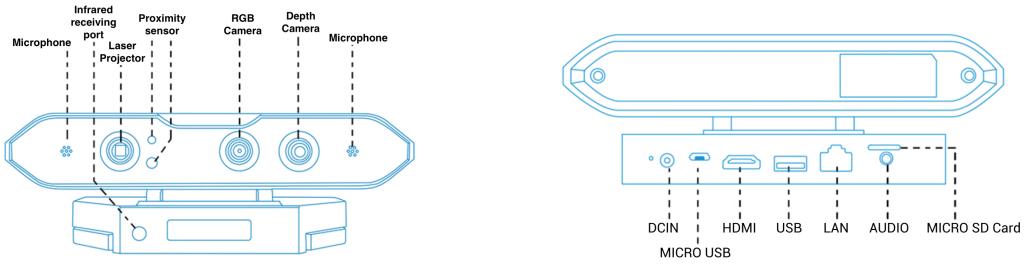
Figure 5.13: Colour and extracted depth images using the rear camera in iPhone X

Table 5.5: Calibration parameters for rear and front camera in iPhone X

	Rear Camera	**Front Camera**
Image size	576×768	480×640
Focal length	1164.2	596.9
Principle points	[382.92 , 285.97]	[320.56 , 237.55]
Pixel size (μm)	1.00	1.22
K1	-0.4113	-0.0565
K2	7.2474	1.568
K3	-32.368	-6.1138
P1	0.0027	-0.00046
P2	-0.0023	0.00037

Both Persee and Astra use the structured light (SL) technique to compute depth. We decided to use Persee because we only needed colour and depth frames for our research. Persee is the worlds first camera-computer. This device can plug into a TV or a monitor, or it can run without a display; we can interact with it entirely through the built-in Astra Pro 3D camera. Persee has a built-in operating system (either Ubuntu or Android). Persee's specifications are shown in Figure 5.15 and Table 5.6. Unfortunately, this device failed before gathering the required frames and so we were not able to utilize it further in this work.

Figure 5.14: Orbbce Persee



(a) (b)

Figure 5.15: (a) Front Persee specification, (b) Backports description

Table 5.6: Persee Specification [10]

	Persee
RGB Resolution	1280×720
Depth Resolution	640×480
Frame rate	30fps
Range	0.6-8m
Field of View	$60°H \times 49°V \times 73°D$

5.7 Conclusion

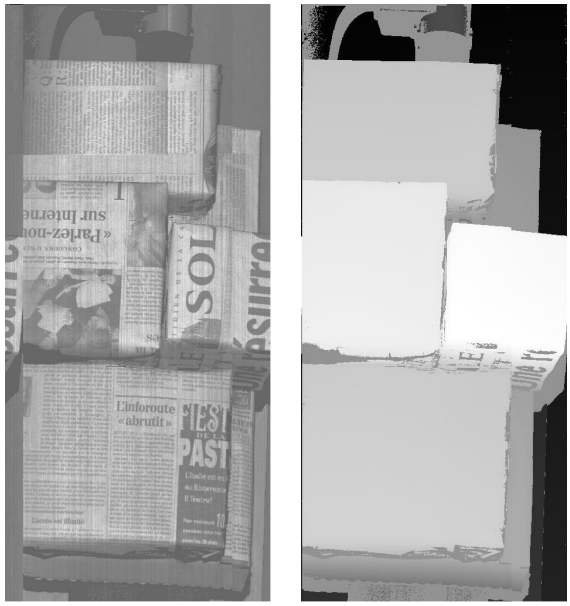
The beginning of the chapter defined 3D imaging techniques that can obtain depth information from the scene. We then described the three sensors used in our research and explain our usage of them. The next chapter expand on the datasets that we used in our research.

Chapter 6

Experimental Datasets

This chapter defines and explains the datasets used to test our algorithms. We show how the correct optical and range flow are computed. The correct 2D/3D flow are shown in both motions translation and divergence.

6.1 NSERC Dataset



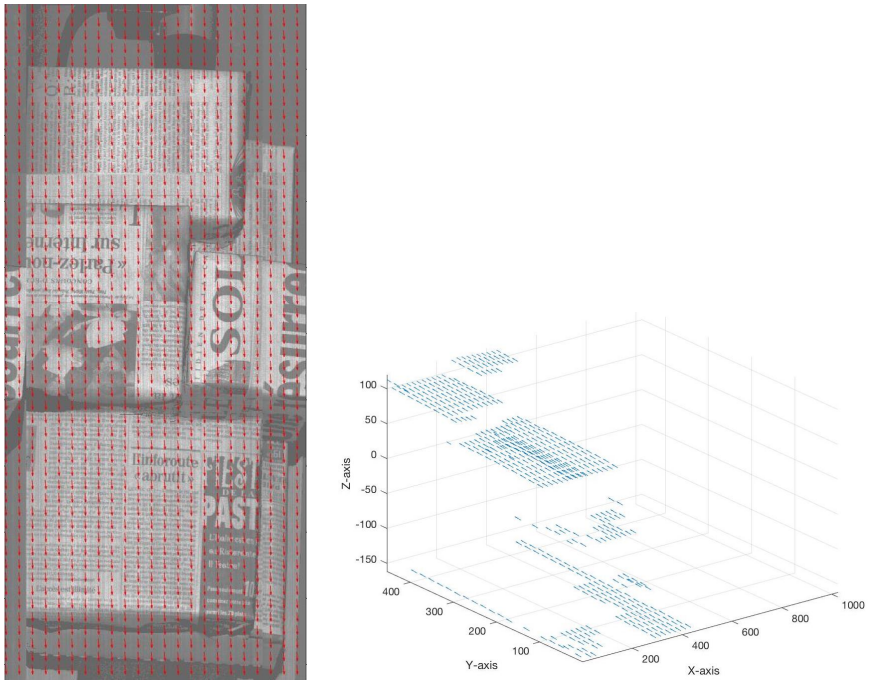
(a) (b)

Figure 6.1: (a) 1st intensity frame in NSERC dataset, and (b) 1st depth frame NSERC dataset

In 1997, Prof.John Barron made a real range sequence using a Biris range sensor at an NSERC lab in Ottawa [98]. The scene consists of some boxes warped in newspaper. Using a linear positioner, the scene moved with a set of fixed equal distances that took range and intensity

images after each movement. Thus, the correct 3D translations are $(0 : 095377; 1 : 424751; 0 : 087113)$ $mm/frame$. The image size of this sequence is 454×1024. Figure 6.1 shows the first intensity and depth frames in the NSERC sequence.

The correct optical flow (u, v) and 3D range flow (u, v, w) are displayed in Figure 6.2.



(a) (b)

Figure 6.2: (a) Correct optical flow (u, v) of the NSERC dataset (b) 3D view of the correct range flow

6.2 Generated New Datasets

As mentioned in the previous chapter, we used the Kinect V2, ZED, and iPhone X to generate RGB-D datasets. Two sequences with two different motions were designed (translation and divergence) for each device. In translation motion, the camera moved $1mm$ to the left; in divergence motion, the camera moved $1mm$ toward the scene. We can compute correct optical flow and range flow for each frame in our data using focal length f, principle point (cx, cy), depth information Z, the amount of translation in millimetres T_x, T_y, T_z, and pixel size per millimetre. For the correct range flow (u, v, w), it is just one millimetre toward the motion direction. However, for correct optical flow, we simply used a motion field equation for the pure translation case to compute (u, v) as the following equations:

$$u_x = \frac{T_z X - T_x f}{Z}, v_y = \frac{T_z Y - T_y f}{Z} \tag{6.1}$$

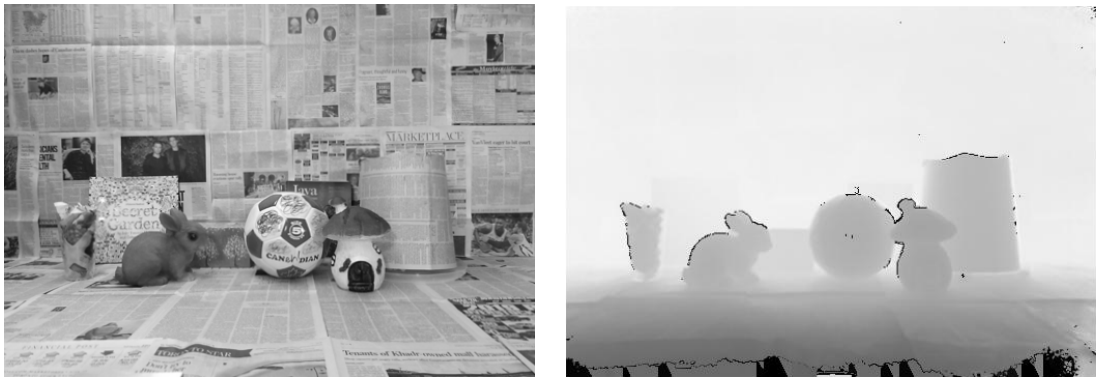
This correct optical/range flow contains some errors due to the inaccurate parameters that we

used to compute the ground-truth flow. Therefore, although we can consider this proper flow as the best estimation for our real data sequences, it is not 100% accurate.

The next section provides further details about the generated datasets for each camera.

6.3 Kinect Dataset

As mentioned in chapter 4, we applied several pre-processing steps to the original Kinect V2 acquired images. Figure 6.3 shows an example for RGB and depth images after the pre-processing steps. The output sequences has the following characteristics, as shown in Table 6.1.



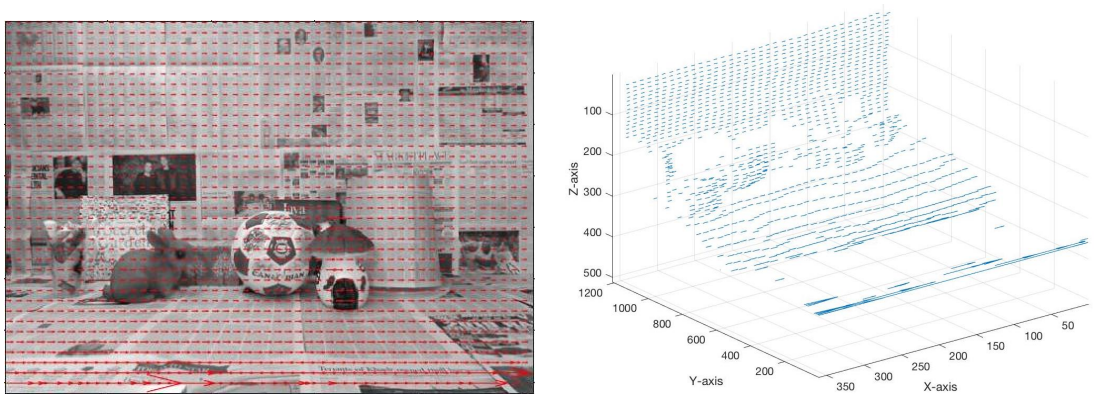
(a) (b)

Figure 6.3: (a) Example of RGB frame in the Kinect sequence, and (b) Example of depth frame in the Kinect sequence.

Table 6.1: Technical details for the Kinect V2 datasets

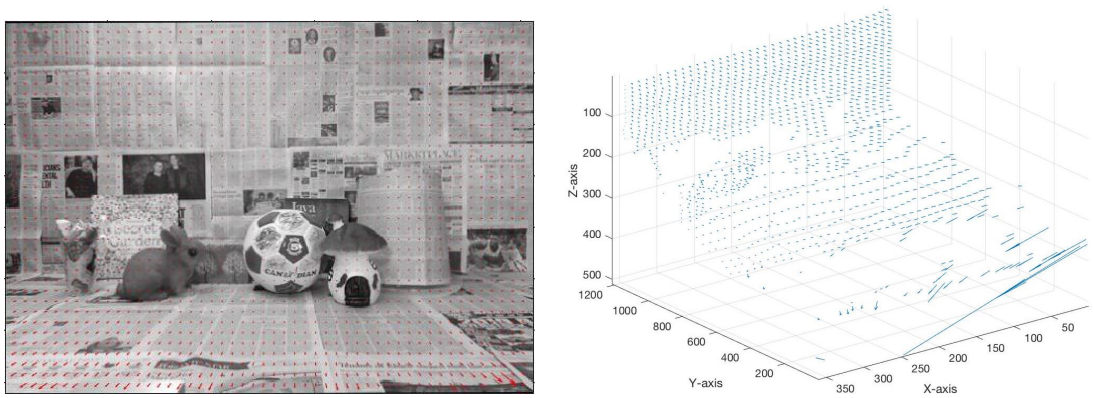
	RGB	**Depth**	**IR**
Image size	512×360	512×360	512×360
Format	8-bit (gray)	16-bit	16-bit
Type	png	png	png
Number of images	20	20	20
Filename	Kinect_Color_Div_%03d Kinect_Color_Trans_%03d	Kinect_Depth_Div_%03d Kinect_Depth_Trans_%03d	Kinect_IR_Div_%03d Kinect_IR_Trans_%03d

The correct 2D optical flow (u, v) and the 3D range flow (u, v, w), using frame 3 and 4 for both translation and divergence motions in Kinect datasets are displayed in Figure 6.4 and Figure 6.5 respectively.



(a) (b)

Figure 6.4: (a) Correct optical flow (u, v) for the translation motion in the Kinect sequence (b) 3D view of the correct range flow.



(a) (b)

Figure 6.5: (a) Correct optical flow (u, v) for the divergence motion in the Kinect sequence (b) 3D view of the correct range flow.

6.4 ZED Dataset

Figure 6.6 provides examples of RGB and depth frames in ZED dataset sequences. These datasets have the following characteristics, as shown in Table 6.2.

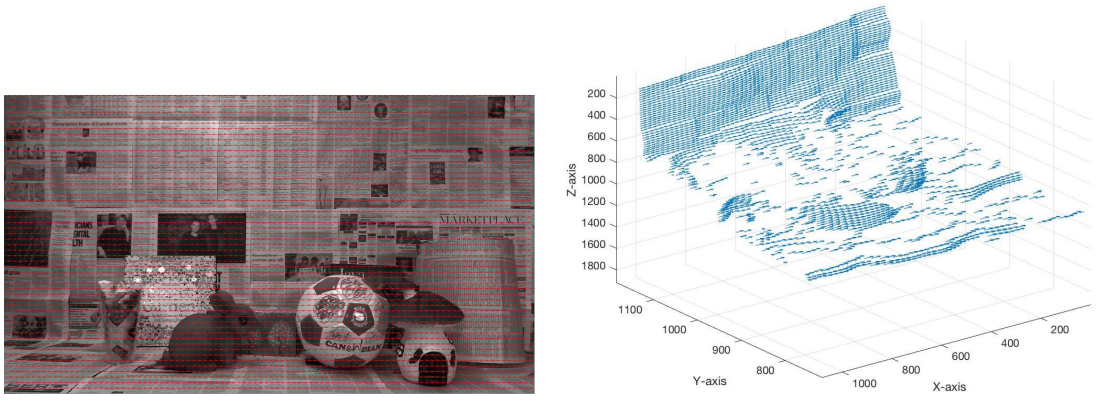
(a) (b)

Figure 6.6: (a) Example of RGB frame in the ZED sequence, and (b) Example of depth frame in the ZED sequence.

Table 6.2: Technical details for the ZED datasets

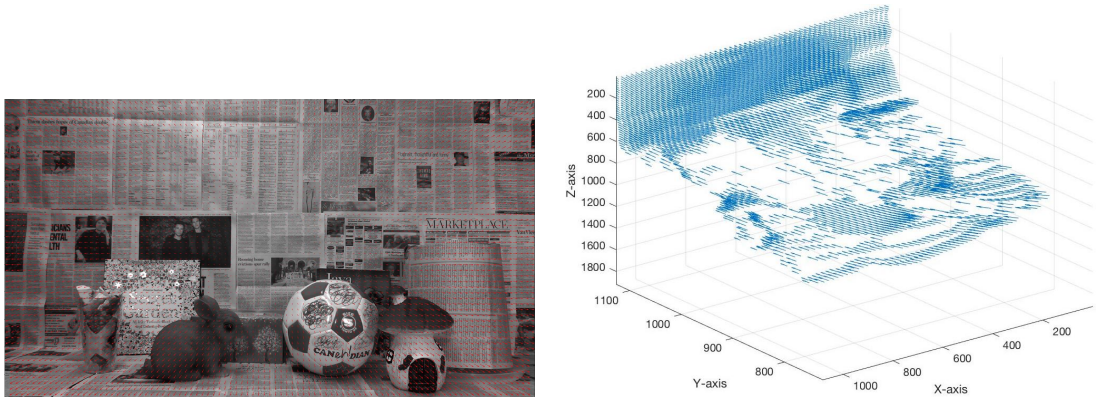
	Left Camera	**Right Camera**	**Depth**
Image size	1080×1920	1080×1920	1080×1920
Format	8-bit(gray)	8-bit(gray)	16-bit
Type	png	png	png
Number of images	20	20	20
Filename	ZED_Left_Div_%03 ZED_Left_Trans_%03d	ZED_Right_Div_%03d ZED_Right_Trans_%03d	ZED_Depth_Div_%03d ZED_Depth_Trans_%03d

The correct 2D optical flow (u, v) and 3D range flow (u, v, w), using frame 3 and 4 for both horizontal translation and divergence motions in ZED datasets are displayed in Figure 6.7 and Figure 6.8 respectively.



(a) (b)

Figure 6.7: (a) Correct optical flow (u, v) for the translation motion in the ZED sequence (b) 3D view of the correct range flow.



(a) (b)

Figure 6.8: (a) Correct optical flow (u, v) for the divergence motion in the ZED sequence (b) 3D view of the correct range flow

6.5 Stereo Camera (rear) in iPhone X Dataset

Examples of RGB and depth images in this dataset are shown in Figure 6.9. Table 6.3 shows the sequence characteristics taken by the rear camera in the iPhone X.

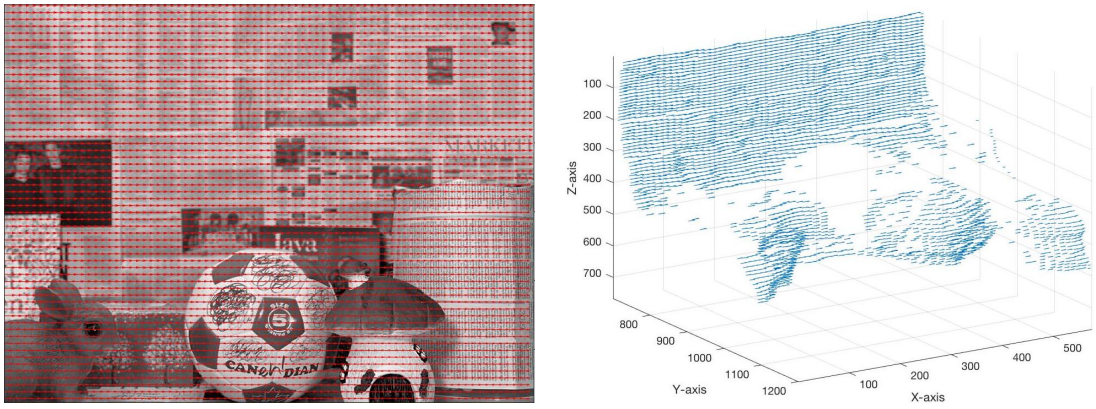
(a) (b)

Figure 6.9: (a) Example of RGB frame in the rear camera sequence, and (b) Example of depth frame in the rear camera of iPhone X.

Table 6.3: Technical details for the rear camera datasets in the iPhone X

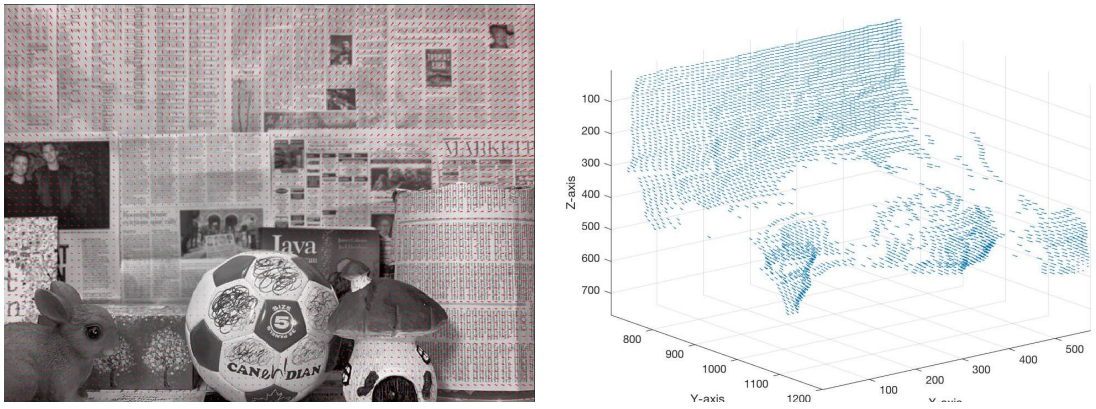
	RGB	**Depth**
Image size	576×768	576×768
Format	8-bit(gray)	16-bit
Type	jpg	png
Number of images	20	20
Filename	Back_Div_Color_%03d Back_Trans_Color_%03d	Back_Div_Depth_%03d Back_Trans_Depth_%03d

The correct 2D optical flow (u, v) and 3D range flow (u, v, w), using frame 3 and 4 for both horizontal translation and divergence motions using rear camera dataset are displayed in Figure 6.10 and Figure 6.11 respectively.



(a) (b)

Figure 6.10: (a) Correct optical flow (u, v) for the translation motion in the rear camera sequence (b) 3D view of the correct range flow.



(a) (b)

Figure 6.11: (a) Correct optical flow (u, v) for the divergence motion in the rear camera sequence (b) 3D view of the correct range flow.

6.6 Truedepth Camera (front) in iPhone X Dataset

Two frames showing RGB and depth as taken by the Truedepth camera are shown in Figure 6.12. Table 6.4 shows front camera dataset characteristics.

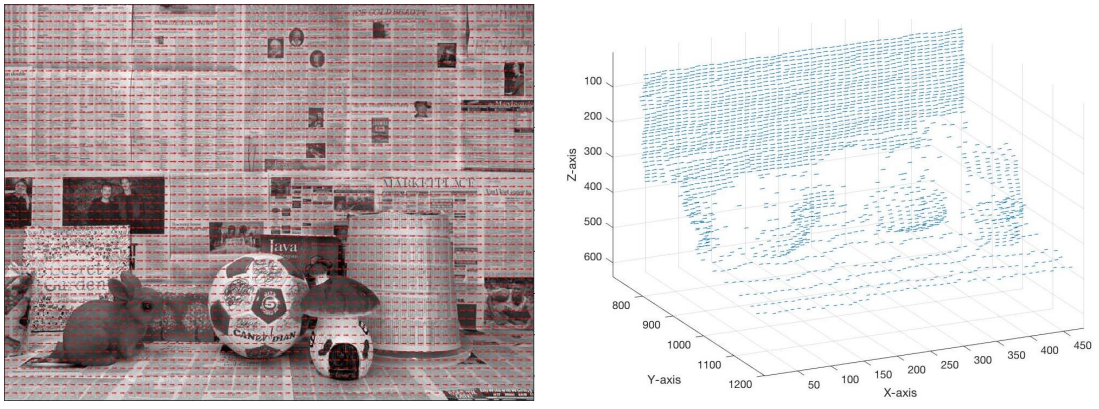
(a) (b)

Figure 6.12: (a) Example of the RGB frame in the front camera sequence, and (b) Example of depth frame in the front camera sequence

Table 6.4: Technical details for the front camera data sets in the iPhone X

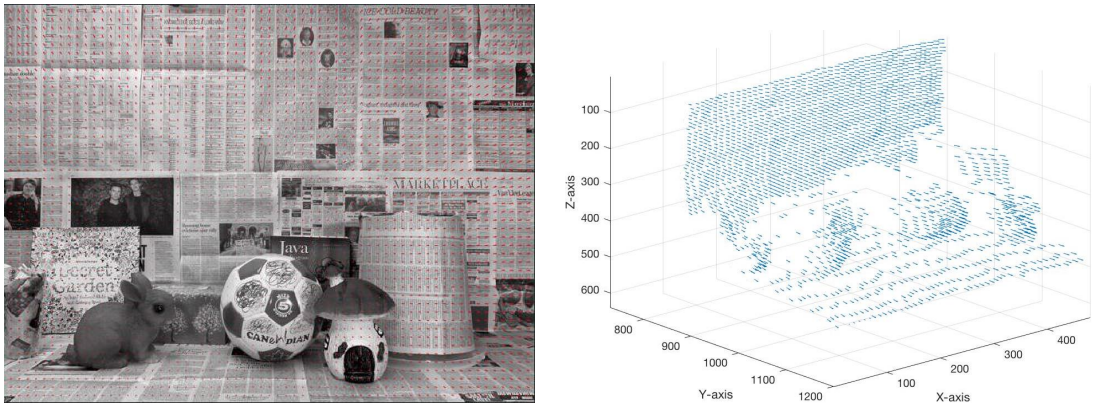
	RGB	**Depth**
Image size	480×640	480×640
Format	8-bit(gray)	16-bit
Type	jpg	png
Number of images	20	20
Filename	Front_Div_Color_%03d Front_Trans_Color_%03d	Front_Div_Depth_%03d Front_Trans_Depth_%03d

The correct 2D optical flow(u, v) and 3D range flow (u, v, w), using frame 3 and 4 for both horizontal translation and divergence motions using Truedepth dataset are displayed in Figure 6.13 and Figure 6.14 respectively.



(a) (b)

Figure 6.13: (a) Correct optical flow (u, v) for the translation motion in the front camera sequence (b) 3D view of the correct range flow



(a) (b)

Figure 6.14: (a) Correct optical flow (u, v) for the divergence motion in the front camera sequence (b) 3D view of the correct range flow

6.7 Middlebury Stereo Datasets

In 2003 [91], Scharstein and Szeliski acquired high-accuracy stereo datasets that has since been used widely in several scene flow methods. They provide colour images and 2 farms of disparity maps. The ground-truth disparity maps were used as the depth channel. The motion in this dataset is equivalent to the ego-motion of the camera in X direction, which coincides with the baseline of the stereo setup, while the motion in Y and Z direction are equal to zero. We used two Middelbury datasets: *Teddy* and *Cones*. The resolution in these frames are 375×450.

Figure 6.15 shows intensity and disparity frames from *Teddy* and *Cones* datasets, plus the correct 2D optical flow.

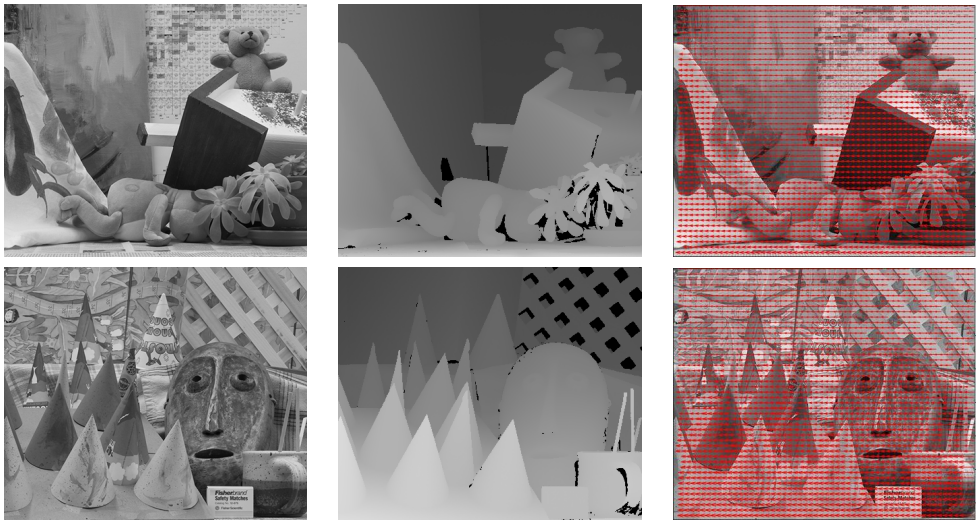


Figure 6.15: Example of Teddy and Cones intensity and disparity frames.

6.8 Conclusion

We explained the datasets used in our research. We showed the correct optical and range flow for each sequence in order to compare them with the estimation flow field. In the next chapter, we provide the experiments using these datasets and analyzing the results.

Chapter 7

Experiments Results and Analysis

This chapter provides experimental results, our quantitative and qualitative analysis, and a discussion of these results. The implemented approaches were tested using a series of datasets, previously described in Chapter 6. For quantitative analysis, we used the error measurements explained in Chapter 4 in section 4.1.13. The first part of this chapter provides a comparison with other methods using Teddy and Cones datasets.

7.1 Middleburry Datasets For Comparison

Middlebury stereo datasets [91] are a common benchmark to compare range/scene flow methods [130, 125, 156, 154, 153, 140, 157, 222]. In this experiment, we obtained gray images from RGB frames and gained depth images from disparity frames using focal length based on formula $Z = (f * b)/d$. As in the comparison studies, we used frames 2 and 6 as the first and second frames, respectively. The ground truth of the scene coincides with the camera motion along the X-axis. Therefore, the Y and Z motions are zero, while X motion is constant. Optical flow was not constant, as shown by the disparity map of the first image. The ground truth and further details about these datasets were previously explained in Chapter 6, section 6.7. Since comparison studies combined intensity and depth images in their regularization framework, we only used regularization approaches that similarly combined intensity and depth data (**Global_ir, Global_ind_ir, CLG_ir, Brox_ir**). Notably, we are going to show that we can obtain 3D range flow using depth information only using (**Global_r, Global_ind_r, CLG_r, Brox_r**) approaches.

Overall, we considered eight comparison studies, previously described in the literature section in Chapter 2. Four studies are **(RGBD)** based methods: Xiang et al. in 2018 [130], Quiroga et al. in 2013 [125], Hadfield and Bowden in 2014 [156], and Sun et al. in 2015 [157]. Three studies are stereo **(ST)** based methods: Huguet and Devernay in 2007 [140], Vogel et al. in 2013 [154], and Basha et al. 2013 [153]. Lastly, Brox and Malik added depth motion **(OP+depth)** to the results of their optical flow study flow [222]. In terms of **RMSE** (Root Mean Squared Errors) as explained in equation (4.22) and 2D **AAE** (Average Angular Errors) using equation (4.16), the quantitative results for *Teddy* and *Cones* datasets are shown in Table

7.1. To get this result, we used the maximum number of levels possible with equation (4.2), which is 57 levels with $\eta = 0.95$.

Table 7.1 shows that the best overall results for **RGBD** or **ST** methods is Sun et al.'s [157] method in both *Teddy* and *Cones* datasets. Our approaches successfully estimated the range flow concerning angle direction. Using Table 7.1, we can conclude that our methods show that we can use RGBD data to obtain reasonable AAE, which is in the same range as other scene flow methods. Importantly, the RMSE is slightly elevated because the Mean Absolute Errors (MAE) are high, as seen in Table 7.2. Although this shows that the output flow is accurate in term of angle direction of the motion, this nevertheless indicates that the magnitude of the flow needs to be improved. Nevertheless, our methods can provide adequate 3D range flow using depth data only. In other words, the RMSE is high because the MAE is elevated, but the AAE is within the acceptable range.

Our *Teddy* dataset results show that the **CLG_ir** approach provides the best results when combining intensity with range data. However, when utilizing range data only, **Brox_r** gives superior results. *Cones* dataset reacted differently, indicating that **Global_ind_ir** performs better when combining intensity with range data while **CLG_r** gives the best estimation of 3D flow using range data.

Table 7.1: Root Mean Squared Error (RMSE) and Average Angular Error (AAE) results on Middlebury dataset

Method	**Teddy datasets**		**Cones datasets**	
	RMSE	**AAE**	**RMSE**	**AAE**
Xiang (RGBD)[130]	0.55	0.62	0.56	0.38
Quiroga (RGBD)[125]	0.94	0.84	0.79	0.52
Hadfield (RGBD)[156]	0.52	1.36	0.59	1.61
Sun (RGBD)[157]	0.09	0.17	0.12	0.13
Vogel (ST)[154]	0.81	0.53	1.16	0.6
Basha (ST)[153]	0.57	1.01	0.58	0.39
Huguet and Devernay (ST) [140]	1.25	0.51	1.1	0.69
Brox and Malik (OP+depth) [222]	2.11	0.43	2.3	0.52
Global_ir (RGBD)	9.91	0.77	12.44	0.68
Global_ind_ir (RGBD)	3.15	0.84	3.3	0.48
CLG_ir (RGBD)	5.02	0.66	6.30	0.52
Brox_ir (RGBD)	9.51	0.85	11.58	0.64
Global_r (Depth)	9.03	1.3	25.42	7.89
Global_ind_r (Depth)	9.19	1.63	12.02	8.26
CLG_r (Depth)	8.9	1.6	10.06	0.85
Brox_r (Depth)	9.34	0.78	11.72	0.93

Error measurements regarding the percentage of the flow density and time are shown in Table 7.2. The output flow densities are not 100%. *Teddy* is (97.98%) and *Cones* is (96.78%) because there are missing spots in the disparity and depth data as seen in Figure 6.15 (Black spots). In

addition, this table also shows the computation of **EPE** (End Point Error) and **MAE** (Mean Absolute Error) as described in Section 4.1.13.

Figure 7.1 and Figure 7.2 show the output flow in a vector mode visualization for *Teddy* and *Cones* datasets, respectively, using regularization approaches.

Table 7.2: Error measurements for ***Teddy*** and ***Cones*** datasets using different methods

Teddy datasets						
Methods	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
Global_ir	97.98	0.77	7.96	8.05	9.91	678.82
Global_ind_ir	97.98	0.84	1.87	2.11	3.15	194.12
CLG_ir	97.98	0.66	3.78	3.96	5.02	588.57
Brox_ir	97.98	0.85	7.76	7.93	9.51	1244.00
Global_r	97.98	1.30	8.23	8.70	9.03	372.83
Global_ind_r	97.98	1.63	7.16	7.61	9.19	679.26
CLG_r	97.98	1.60	7.38	7.99	8.90	59.23
Brox_r	97.98	0.78	8.02	8.12	9.34	372.07
Cones datasets						
Methods	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
Global_ir	96.78	0.68	10.40	10.53	12.44	801.29
Global_ind_ir	96.78	0.48	1.73	1.91	3.30	212.56
CLG_ir	96.78	0.52	4.74	4.90	6.30	583.38
Brox_ir	96.78	0.64	10.34	10.42	11.58	1154.58
Global_r	96.78	7.89	22.64	23.96	25.42	512.39
Global_ind_r	96.78	8.26	8.55	10.33	12.02	973.43
CLG_r	96.78	0.85	8.53	8.85	10.06	71.01
Brox_r	96.78	0.93	10.14	10.49	11.72	527.77

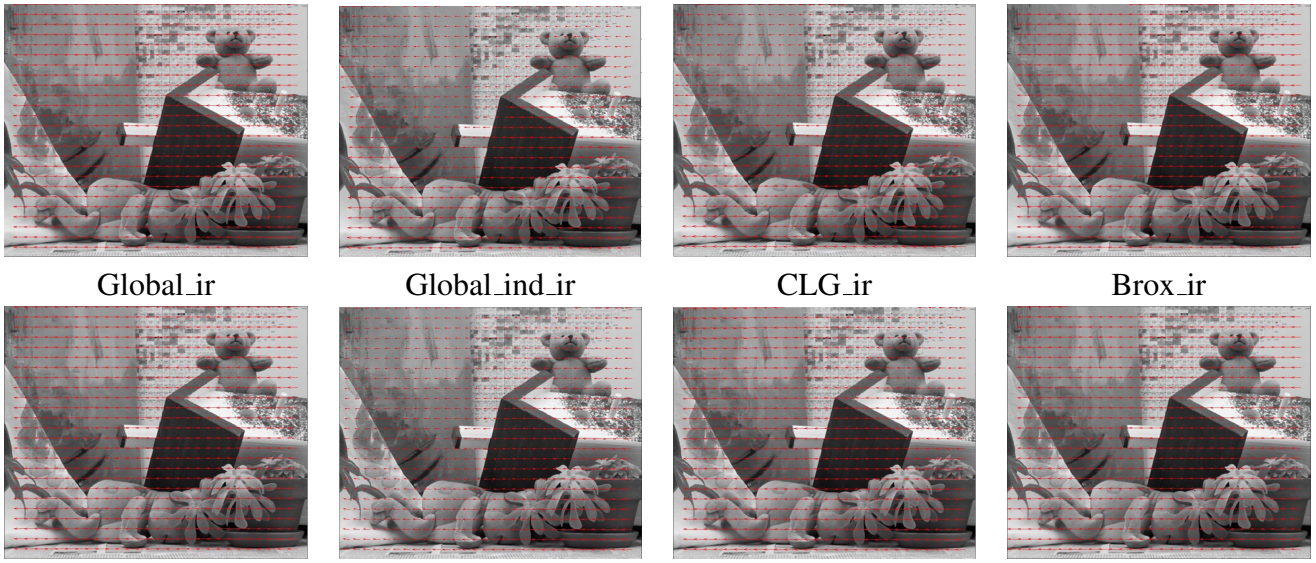


Figure 7.1: Teddy results using the regularization approaches

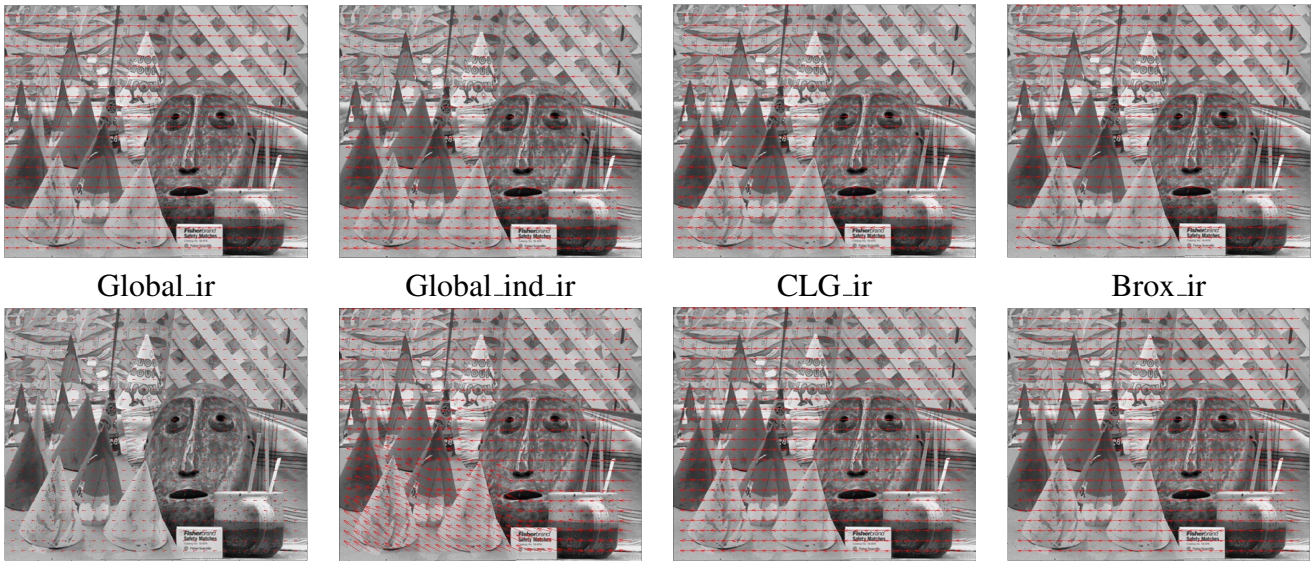


Figure 7.2: Cones results using the regularization approaches

7.2 Experiments Using Generated Datasets

18 approaches need to be tested: six approaches to estimate the 2D optical flow use intensity data only **(LS_i, TLS_i, Global_i, Global_ind_i, CLG_i, Brox_i)**. The other six approaches to compute 3D range flow use range data only **(LS_r, TLS_r, Global_r, Global_ind_r, CLG_r,**

Brox_r). The last six approaches to estimate the 3D range flow use a combination of intensity and depth data **(LS_ir, TLS_ir, Global_ir, Global_ind_ir, CLG_ir, Brox_ir)**. Moreover, we would like to test six different robust functions with the previously mentioned approaches **Brox, Bruhn, Charbonnier, G-Charbonnier, Lorentzian** and **Geman-McClure**. In addition, we also have eight newly generated datasets with two different motions (translation and divergence) to test: **Kinect_Trans, Kinect_Div, ZED_Trans, ZED_Div, Front_Trans, Front_Div, Rear_Trans,** and **Rear_Div**. However, testing all these 18 approaches with six robust functions including the new 8 datasets would produce non-readable results.

Therefore, we provide a few experiments that do demonstrate specific objectives and conclusions. The experiments in the next section demonstrate the effect of using the pyramid in the framework and test its ability to estimate large displacement. In addition, our experiments compare the two local approaches, the two global approaches, the local versus global approaches, to illustrate the difference between**CLG** and **Brox** methods. Furthermore, our experiments also illustrate the difference in estimating 3D range flow using range data (R) only and combining the intensity with range data (IR) together. They also test the six robust functions in the local and global approaches. Further, we provide several small experiments to show the effect of using the weighted term of intensity and range derivatives and weighted median filter for the intermediate flow. Lastly, we estimate the two different motions (translation and divergence) captured by different cameras/sensors.

The output of these experiments is shown in two different visualization modes, as described in section 4.1.12. The vector field mode shows the flow as arrows to represent the direction and the magnitude, while colour mode uses Middlebury colour coding to show the direction and the magnitude of the motion. For example, the red colour represents the right direction while the blue colour represents the left direction.

Lastly, we used the error measurements described in section 4.1.13 to evaluate the estimated flow and provide a quantitative analysis of the results. Using our generated real datasets, we can consider the results acceptable by looking at the visualization and the error measurements equally. Having a consistent visualization flow with low errors (AAE $< 25°$) is an acceptable estimated flow using our new generated datasets.

7.3 Pyramid Effect

To demonstrate the pyramid effect in our approaches, we tested two different datasets: **Kinect_trans** and **ZED_trans** datasets. In these experiments, we applied one local approach, Least Square (**LS**), and one regularized approach, Combined Local and Global (**CLG)**, with three different datatype (**i (intensity only),r (range only),ir (intensity and range together)**).

We tested Gaussian pyramid when ($\eta = 0.5$ with levels number = 4). In addition, we tested the Brox pyramid ($\eta = 0.95$) with different levels $1, 4, 10, max$. max value was computed using equation (4.2), which is equal to 56 levels in the Kinect dataset (with size 360×512) and 64 levels in the ZED dataset (using half the size of the ZED dataset with size 450×960 to reduce the execution time). We present the results in Table 7.3, which shows the percentage of the density of output flow, AAE (Average Angular Error) in degrees, and execution time in

seconds. In 2D approaches (when using intensity only), 2D errors were computed. 3D flow (when using range or combined intensity with range) computed 3D errors.

We found that the output flow results using the pyramid improved when using more than one level of the image without the pyramid. The Gaussian pyramid produced good results like the Brox pyramid. However, although using the Brox pyramid with ten levels produced similar results to the Gaussian pyramid, it did required more time. In other words, although using a *max* number of levels gives the best results, it does require a longer execution time. In the local approaches (LS), we increased the percentage of the density of the output flow by increasing the level number in the pyramid. With the *max* number of levels, we can obtain 100% of the flow.

Table 7.3: Error measurements to demonstrate the pyramid effect

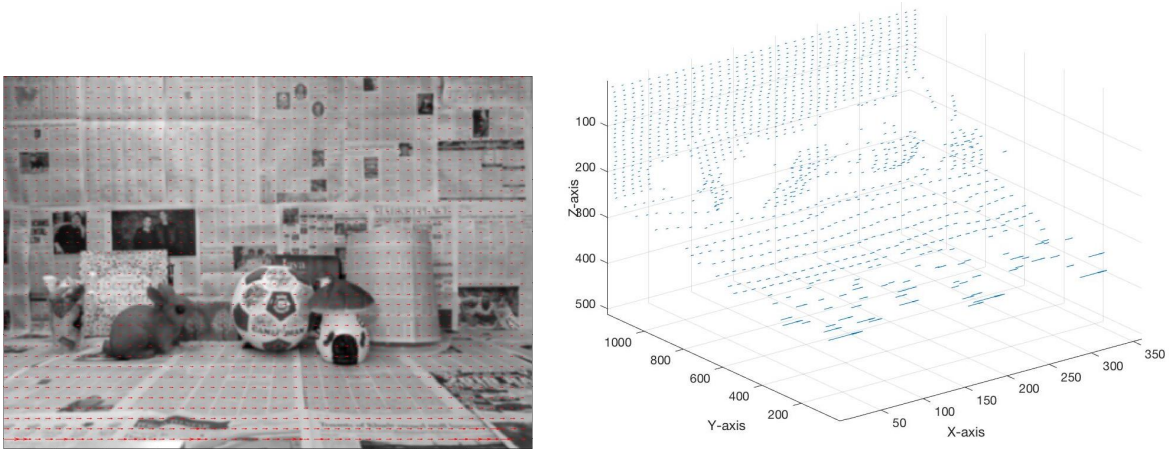
			Kinect_Trans			**ZED_Trans**		
Method	**Levels**	η	**Density**	**AAE**	**Time**	**Density**	**AAE**	**Time**
LS_i	1		93.85	9.45	18.74	98.53	13.20	45.26
	4	0.95	97.1	9.19	51.78	99.53	12.45	143.3
	10		99.34	8.87	109.26	99.85	12.14	268.76
	max		100	8.53	195.25	100	11.88	387.15
	4	0.5	100	8.53	26.39	99.29	11.78	52.5
LS_r	1		35.03	23.31	14.49	2.02	40.58	32.03
	4	0.95	63.38	25.55	50.8	48.2	28.23	107.83
	10		78.24	26.45	92.34	33.8	26.18	234.18
	max		100	32.99	97.89	100	32.07	297.05
	4	0.5	65.13	23.08	21.18	98.33	32.03	38.47
LS_ir	1		97.01	19.13	10.16	73.47	17.33	37.08
	4	0.95	98.99	20.95	33.56	91.88	16.48	112.64
	10		99.24	21.36	75.05	99.17	18.69	199.03
	max		100	23.59	101.28	100	19.47	320.17
	4	0.5	97.7	19.25	15.66	95.85	19.80	47.93
CLG_i	1		100	10.41	7.71	100	16.04	28.63
	4	0.95	100	9.16	16.78	100	12.66	60.85
	10		100	8.74	28.53	100	11.21	94.26
	max		100	8.76	46.34	100	11.08	106.65
	4	0.5	100	8.54	6.75	100	10.42	18.23
CLG_r	1		100	25.60	13.96	99.84	42.69	73.22
	4	0.95	100	21.55	51.35	100	41.11	238.8
	10		100	18.77	107.66	100	37.36	343.28
	max		100	18.95	149.38	100	14.91	164.42
	4	0.5	100	10.93	24.6	100	27.84	28.96
CLG_ir	1		100	18.81	18.66	100	30.87	51.94
	4	0.95	100	12.27	58.42	100	18.49	174.54
	10		100	10.76	108.46	100	12.76	212.35

...continued

			Kinect_Trans			ZED_Trans		
Method	**Levels**	η	**Density**	**AAE**	**Time**	**Density**	**AAE**	**Time**
	max		100	13.65	152.04	100	13.78	243.57
	4	0.5	100	9.19	21.27	100	11.02	50.99

7.4 Large Displacement Estimation

The estimated the ability to handle large displacements using 18 approaches is tested. This test was done without any robust methods. We used **Kinect_trans** datasets in this experiment with differring numbers of frames. As mentioned in Chapter 5, images were acquired every 1*mm*. Twenty images were captured for a total movement of 2*cm*. In our experiments, we used the Kinect intensity (grayvalue) images, and the depth frames with size 512×360. In testing all 18 approaches, we used four levels with $\eta = 0.5$ to build the Gaussian pyramid. We choose to start with frame 5. The best estimation of the correct flow (ground truth flow) using frame 5 and 6 are shown in Figure 7.3 as a vector representation and 3D plot.



Vector Flow 3D Flow

Figure 7.3: Correct flow for the Kinect translation motion using frame 5 and 6

For each test, we needed four frames as described in Section 4.1.1. In this experiment, we defined five different steps of motion as follows. In step one, we read frames $(5, 6, 7, 8)$, which had 1*mm* between each frame. Conversely, in step 5, we used $(5, 10, 15, 20)$ frames with 5*mm* displacement. Similarly, step 2 required $(5, 7, 9, 11)$, step 3 needed $(5, 8, 11, 14)$, and step 4 read $(5, 9, 13, 17)$. In general, we tested the ability to handle up to 5*mm* large motion. Table 7.4 shows the results of these experiments for the each of the implemented 18 approaches. In the 2D optical flow approaches (when applied intensity data only), 2D errors are computed; for the 3D range flow, 3D errors are estimated.

Table 7.4 shows that all of our 18 approaches successfully estimated the 2D and 3D range flow of the Kinect translation motion. Since we cannot be sure that the correct flow is 100% correct (it is our best estimation for the ground truth), the error measurements in this table can be considered acceptable results. We estimated the errors in these experiments using the percentage of the density of the output flow, AAE (Average Angular Errors), EPE (End Point Error), MAE (Mean Absolute Error), and RMSE (Root Mean Squared Error).

In 2D optical flow approaches **LS_i, TLS_i, Global_i, Global_ind_i, CLG_i** and **Brox_i**, the large displacement can be estimated successfully even with the 5*mm* displacement in the fifth step. Using range data only to estimate the 3D range flow with these approaches **LS_r, TLS_r, Global_r** and **GLobal_ind_r** were acceptable for the first step only. Larger displacements produced big errors. (We considered errors to be big when the AAE > 25°). However,combining intensity and range data approaches **LS_ir, TLS_ir, Global_ir** and **GLobal_ind_ir** did produce satisfactory results. In contrast, the CLG and Brox approaches estimated the 3D range flow successfully, even with the large displacement 5*mm* in the fifth step, using either the combination between intensity and range data or isolating only the range data. **CLG_r, CLG_ir, Brox_r** and **Brox_ir** approaches produced the best 3D range flow compared with other results.

Table 7.4: Error measurements to demonstrate the handling of the large motion

Method	Step	Frames	Density	AAE	EPE	MAE	RMSE	Time
LS_i	1	5,6,7,8	100	8.53	0.21	0.23	0.43	28.98
	2	5,7,9,11	100	11.72	0.46	0.49	0.98	26.17
	3	5,8,11,14	100	11.32	0.71	0.75	1.83	24.16
	4	5,9,13,17	100	10.88	0.89	0.99	1.99	27.05
	5	5,10,15,20	100	8.99	1.36	1.44	11.28	22.97
LS_r	1	5,6,7,8	65.13	23.08	0.46	0.55	0.48	19.05
	2	5,7,9,11	49.97	37.22	0.90	1.01	0.93	19.64
	3	5,8,11,14	30.9	49.11	1.38	1.51	1.41	17.49
	4	5,9,13,17	27.82	57.83	1.90	2.08	1.93	17.47
	5	5,10,15,20	28.63	63.85	2.39	2.64	2.47	16.88
LS_ir	1	5,6,7,8	98.73	18.59	0.62	0.75	2.00	14.25
	2	5,7,9,11	99.13	21.60	0.82	1.04	1.83	12.88
	3	5,8,11,14	99.13	21.33	1.14	1.44	2.78	12.15
	4	5,9,13,17	98.57	23.58	1.60	2.05	3.55	16.59
	5	5,10,15,20	98.81	20.62	1.75	2.24	4.59	13.66
TLS_i	1	5,6,7,8	100	8.48	0.21	0.23	0.44	26.82
	2	5,7,9,11	100	11.83	0.46	0.50	0.99	25.33
	3	5,8,11,14	100	11.56	0.72	0.77	1.85	25.24
	4	5,9,13,17	99.59	12.86	0.97	1.05	2.03	21.66
	5	5,10,15,20	100	9.09	1.38	1.45	11.30	23.35
TLS_r	1	5,6,7,8	70.78	26.49	0.55	0.69	0.64	15.62
	2	5,7,9,11	75.13	39.37	0.98	1.20	1.07	17.04
	3	5,8,11,14	64.96	48.07	1.44	1.73	1.55	15.31

. . . continued

Method	Step	Frames	Density	AAE	EPE	MAE	RMSE	Time
	4	5,9,13,17	58.25	54.60	1.91	2.25	1.99	14.92
	5	5,10,15,20	54.03	60.28	2.39	2.79	2.46	14.67
TLS_ir	1	5,6,7,8	98.42	9.56	0.22	0.26	0.40	13.62
	2	5,7,9,11	99.17	14.34	0.51	0.61	1.02	12.59
	3	5,8,11,14	99.53	14.18	0.83	0.94	2.09	9.8
	4	5,9,13,17	98.93	18.75	1.26	1.37	2.37	14.16
	5	5,10,15,20	99.47	13.83	1.65	1.78	11.43	11.72
Global_i	1	5,6,7,8	100	8.73	0.22	0.24	0.44	8.28
	2	5,7,9,11	100	11.48	0.45	0.49	0.97	5.54
	3	5,8,11,14	100	11.11	0.70	0.75	1.82	4.61
	4	5,9,13,17	100	11.08	0.92	1.01	2.00	9.12
	5	5,10,15,20	100	8.92	1.36	1.44	11.27	5.45
Global_r	1	5,6,7,8	100	17.01	0.45	0.63	1.10	3.3
	2	5,7,9,11	100	27.65	0.85	1.09	1.45	3.73
	3	5,8,11,14	100	30.98	1.27	1.55	2.31	3.94
	4	5,9,13,17	100	30.29	1.57	1.97	2.68	3.53
	5	5,10,15,20	100	33.25	2.31	2.79	11.59	4.04
Global_ir	1	5,6,7,8	100	15.97	0.42	0.56	0.94	3.47
	2	5,7,9,11	100	17.07	0.59	0.80	1.20	3.14
	3	5,8,11,14	100	18.41	0.91	1.22	2.14	2.74
	4	5,9,13,17	100	22.46	1.33	1.78	2.52	6.18
	5	5,10,15,20	100	19.27	1.70	2.22	11.39	3.59
Global_ind_i	1	5,6,7,8	100	8.52	0.21	0.23	0.43	26.25
	2	5,7,9,11	100	11.49	0.46	0.49	0.97	25.99
	3	5,8,11,14	100	11.18	0.71	0.76	1.83	25.63
	4	5,9,13,17	100	10.45	0.91	0.99	1.99	25.13
	5	5,10,15,20	100	8.99	1.38	1.46	11.28	24.89
Global_ind_r	1	5,6,7,8	100	21.36	0.46	0.56	0.62	29.41
	2	5,7,9,11	100	36.46	1.01	1.14	1.39	28.15
	3	5,8,11,14	100	50.45	1.71	1.82	2.50	26.45
	4	5,9,13,17	100	58.58	2.33	2.46	3.06	25.7
	5	5,10,15,20	100	62.80	3.18	3.37	11.72	25.69
Global_ind_ir	1	5,6,7,8	100	14.64	0.35	0.46	0.60	27.49
	2	5,7,9,11	100	17.36	0.59	0.78	1.08	27.85
	3	5,8,11,14	100	17.90	0.89	1.15	1.96	27.95
	4	5,9,13,17	100	20.21	1.23	1.65	2.31	27.38
	5	5,10,15,20	100	18.01	1.65	2.12	11.34	28.15
CLG_i	1	5,6,7,8	100	8.53	0.22	0.23	0.44	13.21
	2	5,7,9,11	100	11.33	0.46	0.49	0.99	13.58
	3	5,8,11,14	100	11.09	0.72	0.77	1.86	13.79

... continued

Method	Step	Frames	Density	AAE	EPE	MAE	RMSE	Time
	4	5,9,13,17	100	10.05	0.94	1.00	2.06	13.15
	5	5,10,15,20	100	8.99	1.43	1.50	11.31	13.73
CLG_r	1	5,6,7,8	100	11.09	0.26	0.32	0.50	16.27
	2	5,7,9,11	100	20.57	0.69	0.77	1.21	15.96
	3	5,8,11,14	100	22.79	1.09	1.19	2.17	16.52
	4	5,9,13,17	100	20.80	1.40	1.52	2.48	16.09
	5	5,10,15,20	100	20.18	2.05	2.20	11.51	16.01
CLG_ir	1	5,6,7,8	100	8.94	0.22	0.26	0.44	15.94
	2	5,7,9,11	100	11.58	0.46	0.51	0.99	17.15
	3	5,8,11,14	100	11.11	0.71	0.78	1.86	17.44
	4	5,9,13,17	100	9.93	0.91	1.01	2.04	16.41
	5	5,10,15,20	100	8.84	1.38	1.49	11.30	18.75
Brox_i	1	5,6,7,8	100	8.47	0.21	0.23	0.44	14.45
	2	5,7,9,11	100	11.39	0.45	0.49	0.98	15.16
	3	5,8,11,14	100	11.00	0.71	0.75	1.83	14.72
	4	5,9,13,17	100	9.73	0.89	0.96	2.00	17.25
	5	5,10,15,20	100	8.72	1.37	1.45	11.28	12.6
Brox_r	1	5,6,7,8	100	15.28	0.34	0.43	0.56	14.52
	2	5,7,9,11	100	21.11	0.69	0.78	1.17	14.79
	3	5,8,11,14	100	21.73	1.05	1.17	2.07	15.2
	4	5,9,13,17	100	21.43	1.37	1.55	2.38	13.13
	5	5,10,15,20	100	20.10	1.98	2.15	11.44	13.63
Brox_ir	1	5,6,7,8	100	8.95	0.22	0.27	0.45	14.74
	2	5,7,9,11	100	11.60	0.46	0.51	0.98	14.45
	3	5,8,11,14	100	11.26	0.72	0.80	1.84	16.05
	4	5,9,13,17	100	10.12	0.91	1.04	2.02	15.08
	5	5,10,15,20	100	8.94	1.38	1.52	11.29	14.28

7.5 Local Approaches Comparison

2D optical flow and 3D range flow can be estimated using the local approaches Least Square **(LS)** and Total Least Square **(TLS)**. To demonstrate the difference between **LS** and **TLS**, we estimated 2D optical flow and 3D range flow using intensity data, range data, and combined intensity and range data. For this test, we invoked **LS_i, LS_r, LS_ir, TLS_i, TLS_r** and **TLS_ir** approaches. This test was done without any robust methods. In addition, we used **Kinect_trans** datasets in this experiment. We also used frames 5 and 6. The best estimation of the correct flow (ground truth flow) using frame 5 and 6 are shown in Figure 7.3 as a vector representation and 3D plot. We utilized four levels with $\eta = 0.5$ to build the Gaussian pyramid.

In the **LS** approaches, one threshold value is necessary; however, the **TLS** approaches require two threshold values for 2D optical flow and three thresholds values for the 3D range flow. In this experiment, we used $\tau = 10$ with **LS_i**, $\tau = 1$ with **LS_r** and **LS_ir**. For **TLS_i** we used $\tau 1 = 20, \tau 2 = 20$, **TLS_r** utilized $\tau 1 = 30, \tau 2 = 10, \tau 3 = 1$. Lastly, **TLS_ir** used $\tau 1 = 30, \tau 2 = 10, \tau 3 = 10$. The error measurements for the local approaches using these thresholds is shown in Table 7.5.

Table 7.5 shows the error measurements in terms of the percentage of the flow, AAE (Average Angular Error), EPE (End Point Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and time in seconds.

LS_i and **TLS_i** can estimate similar 2D optical flow. In both approaches, we can obtain 100% dense flow with $AAE = 8.5°$. In 3D range flow, **LS** estimated slightly better 3D flow than **TLS** approach. However, it is clear that the combined approaches of **LS_ir** and **TLS_ir** provide better 3D flow than **LS_r** and **TLS_r** approaches.

Table 7.5: Error measurements to demonstrate the local approaches differences

Method	Density	AAE	EPE	MAE	RMSE	Time
LS_i	100	8.53	0.21	0.23	0.43	26.87
LS_r	65.13	23.08	0.46	0.55	0.48	20.87
LS_ir	98.73	18.59	0.62	0.75	2.00	13.00
TLS_i	100	8.48	0.21	0.23	0.44	35.30
TLS_r	70.78	26.49	0.55	0.69	0.64	16.88
TLS_ir	71.5	11.70	0.24	0.29	0.26	19.55

Figure 7.4 shows the flow estimated using the Least Square approaches **LS**. 2D optical flow is shown in a 2D vector and color representation. 3D range flow is shown as a 2D vector and in a 3D plot for (u,v,w) flow. Figure 7.5 shows the flow estimated using Total Least Square methods **TLS**.

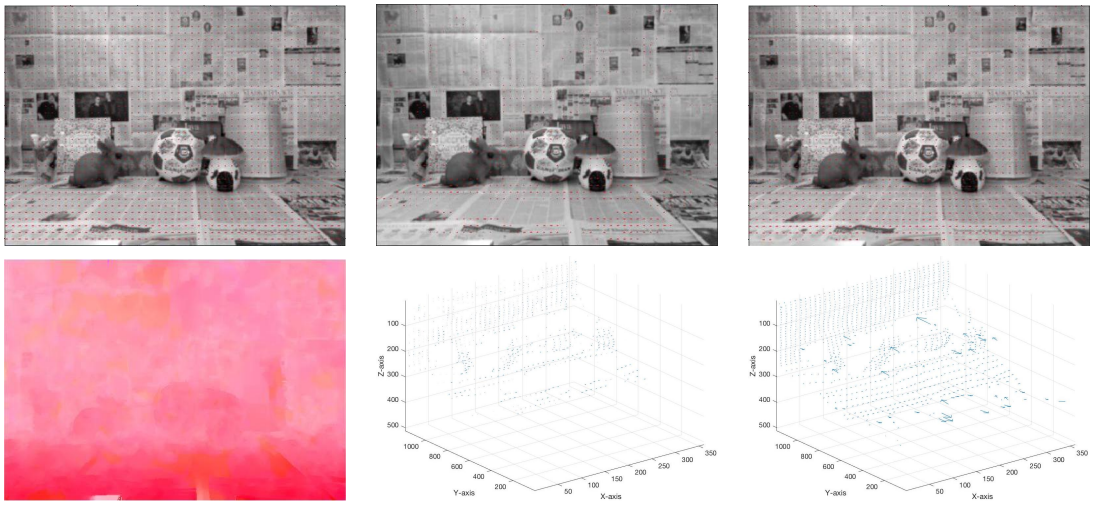


Figure 7.4: Least Square flow using Kinect translation motion

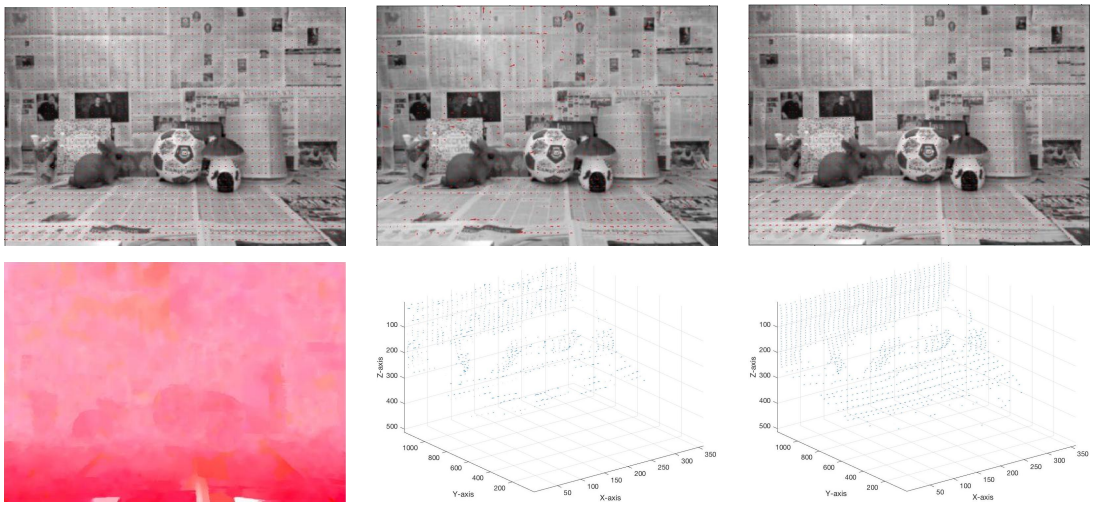


Figure 7.5: Total Least Square flow using Kinect translation motion

7.6 Global Approaches Comparison

This experiment compares the **direct global** approaches and **indirect global** approaches. We estimated 2D optical flow and 3D range flow using intensity data, range data, and the combination of intensity and range data. We invoked **Global_i, Global_r, Global_ir, Global_ind_i, Global_ind_r** and **Global_ind_ir** approaches in this experiment. Similar to the previous test, this experiment was done without using any robust methods. In addition, we applied **Kinect_trans** datasets. We again used frames 5 and 6. The best estimation of the correct flow (ground truth

flow) using frames 5 and 6 is shown in Figure 7.3 as a vector representation and 3D plot. We utilized four levels with $\eta = 0.5$ to build the Gaussian pyramid.

Direct global approaches applied the derivative and regularization on the reading images directly, then estimated the 2D/3D flow. On the other hand, the indirect global approaches computed eigenvalues and eigenvectors with the Least Square methods (**LS**) and subsequently regularized it to estimate dense 2D/3D flow.

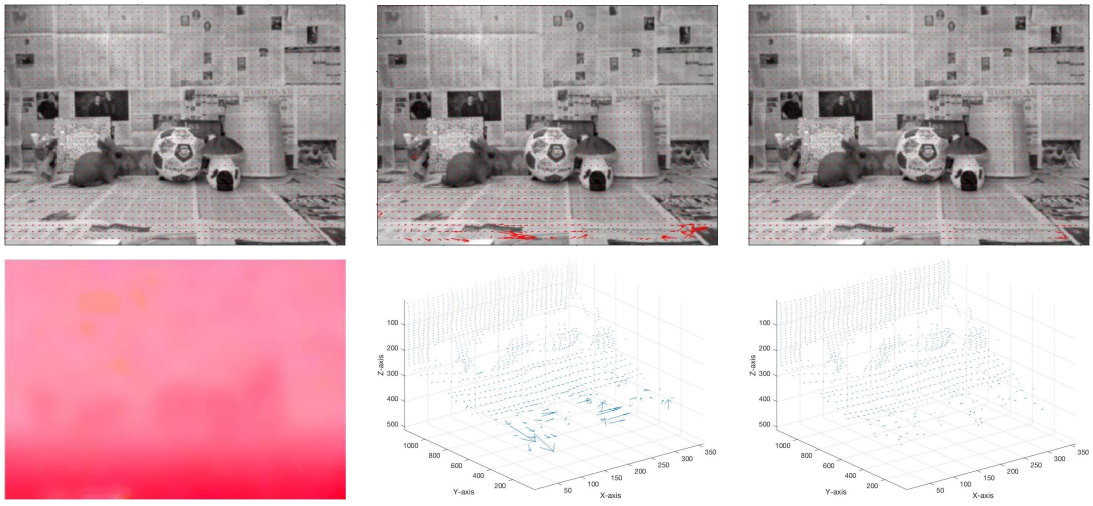
Table 7.6 shows the error measurements to demonstrate the difference between direct and indirect global approaches. This table shows the percentage of the density of the output flow, AAE (Average Angular Error), EPE (End Point Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and the time in seconds.

Using direct and indirect global approaches, we can estimate 100% dense flow in both 2D optical flow and 3D range flow. The AAE is better in the direct global approaches; however, the RMSE is superior in the indirect global approaches. Overall, although indirect approaches require more time because of the local estimation step, they nevertheless produce good flow. Specifically, the output flow using indirect approaches are more robust and do not have outliers like **Global_r** approach. Moreover, estimating the 3D range flow using the combination of intensity and range data improves the output flow compared to using depth information only.

Table 7.6: Error measurements to demonstrate the global approaches differences

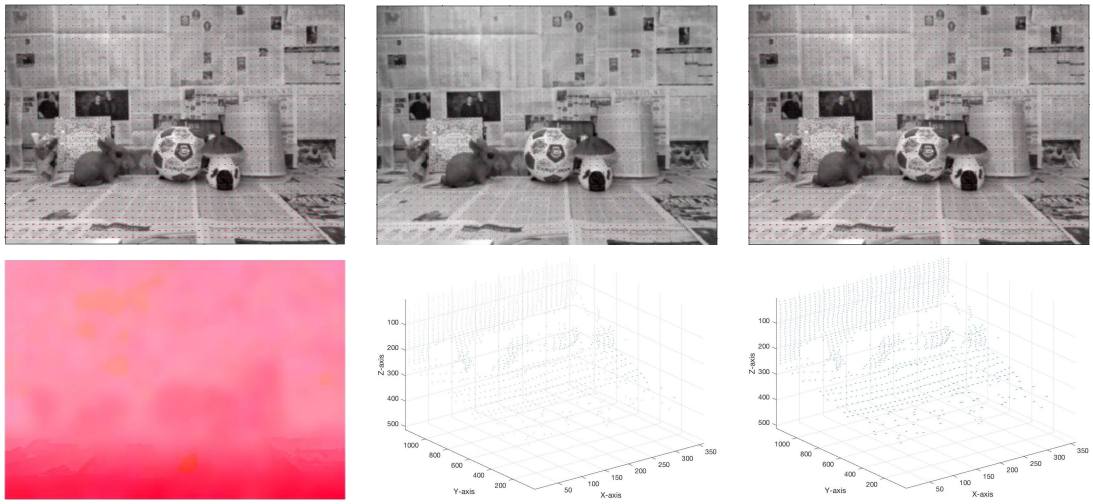
Method	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
Global_i	100	8.75	0.22	0.24	0.44	7.57
Global_r	100	17.01	0.45	0.63	1.10	4.15
Global_ir	100	11.70	0.34	0.47	1.10	2.28
Global_ind_i	100	8.52	0.21	0.23	0.43	25.72
Global_ind_r	100	21.36	0.46	0.56	0.62	34.68
Global_ind_ir	100	16.00	0.39	0.53	0.72	26.74

Figure 7.6 shows the flow estimated using direct global approaches **Global**. 2D optical flow is shown in a 2D vector and color representation. 3D range flow is shown as a 2D vector and in a 3D plot for (u,v,w) flow. Figure 7.7 shows the flow estimated using indirect global methods **Global_ind**.



Global_i Global_r Global_ir

Figure 7.6: Direct Regularization flow using Kinect translation motion



Global_ind_i Global_ind_r Global_ind_ir

Figure 7.7: Indirect Regularization flow using Kinect translation motion

7.7 Local vs. Global Approaches

To demonstrate the difference between local approaches (**LS, TLS**) and global approaches (**direct global, indirect global**), we need to re-examine Table 7.5 and Table 7.6 in addition to Figure 7.4, Figure 7.5, Figure 7.6, and Figure 7.7. In the previous two tests, we used the same **Kinect_trans** datasets with the same settings. Therefore, we are able to demonstrate the difference between local and global results.

Global approaches can estimate 100% dense flow in whole cases, while local approaches cannot estimate 100% of the 3D range flow. Estimating 2D optical flow using local and global

approaches do not produce a significant difference in term of error measurements; however, the 3D range flow estimation is significantly improved. The AAE, for example, is better when estimating flow using global methods compared to local methods. Additionally, indirect global approaches improve the flow estimation using the least squared methods. Therefore, the flow becomes denser and robust in this case. Lastly, direct global methods are much quicker than the local approaches. This is because the indirect approaches require more time during the first step of this method, which must occur prior to regularization.

7.8 CLG and Brox Approaches

Using Bruhn et al.'s method (**CLG**) [4, 5] and Brox et al.'s method (**Brox**) [6], we can estimate the 2D optical flow and 3D range flow as described in Chapter 3. To demonstrate the effect of these methods, we estimated 2D optical flow and 3D range flow using the intensity data **(i)**, range data **(r)**, and the combination of intensity and range data **(ir)**. This time, we executed **CLG_i, CLG_r, CLG_ir, Brox_i, Brox_r** and **Brox_ir** approaches. In the CLG method, Bruhn's robust methods are used as described in their algorithm. Similarly, we also applied the Brox robust techniques as described in their proposed algorithm. We used same datasets in this test as the previous tests (**Kinect_trans**) using frames 5 and 6. Finally, we utilized four levels with $\eta = 0.5$ to build the Gaussian pyramid.

Table 7.7 shows the errors measurements for the CLG and Brox approaches in terms of the dense percentage of the flow, AAE (Average Angular Error), EPE (End Point Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and execution time in seconds.

Using CLG and Brox methods, we can obtain 100% dense of the flow. Both methods produce similar results in terms of error measurements. The AAE, EPE, MAE, and RMSE are very close in **CLG_i** and **Brox_i** approaches. In **CLG_r** and **Brox_r**, the errors are almost identical. Similarly, **CLG_ir** and **Brox_ir** also have comparable results. Both **CLG** and **Brox** estimate 3D range flow better than the previous local and global approaches in term of AAE and RMSE. Moreover, we gained the best 3D range flow for the Kinect_trans datasets using a combination of intensity and range data (**ir**) using **CLG** and **Brox** methods.

Figure 7.8 shows the flow estimated using the Bruhn et al. method **CLG**. 2D optical flow is shown in a 2D vector and color representation. 3D range flow is shown as a 2D vector and in a 3D plot for (u,v,w) flow. Figure 7.9 shows the estimated flow using Brox et al's methods **Brox**.

Table 7.7: Error measurements for CLG and Brox approaches

Method	Density	AAE	EPE	MAE	RMSE	Time
CLG_i	100	8.53	0.22	0.23	0.44	13.37
CLG_r	100	11.09	0.26	0.32	0.50	14.34
CLG_ir	100	8.94	0.22	0.26	0.44	14.55
Brox_i	100	8.47	0.21	0.23	0.44	12.24
Brox_r	100	11.40	0.27	0.31	0.53	15.33
Brox_ir	100	8.95	0.22	0.27	0.45	13.98

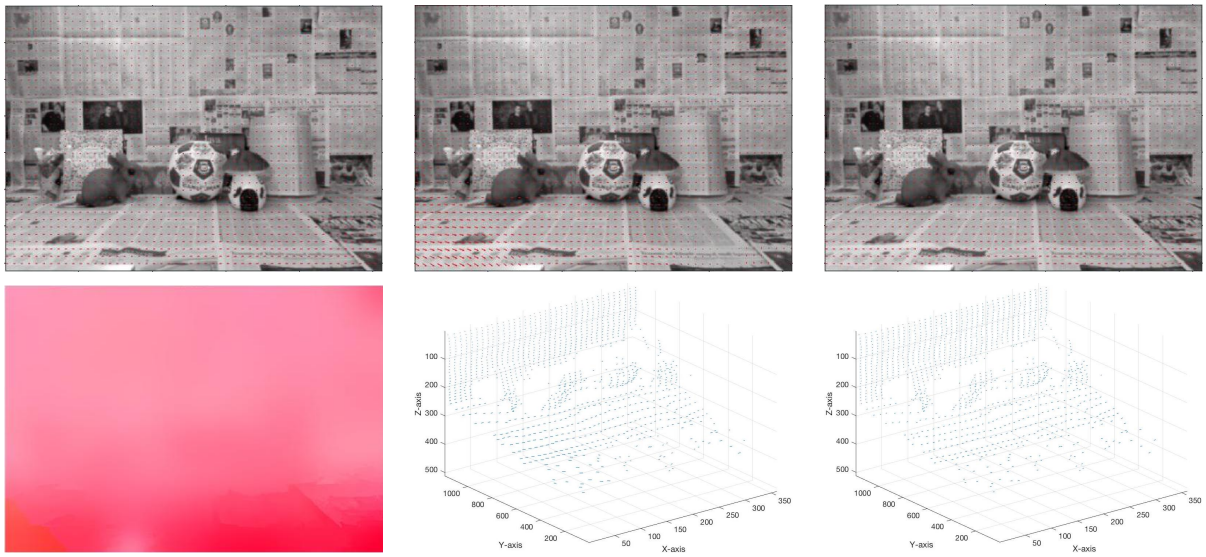


Figure 7.8: CLG Regularization flow using Kinect translation motion

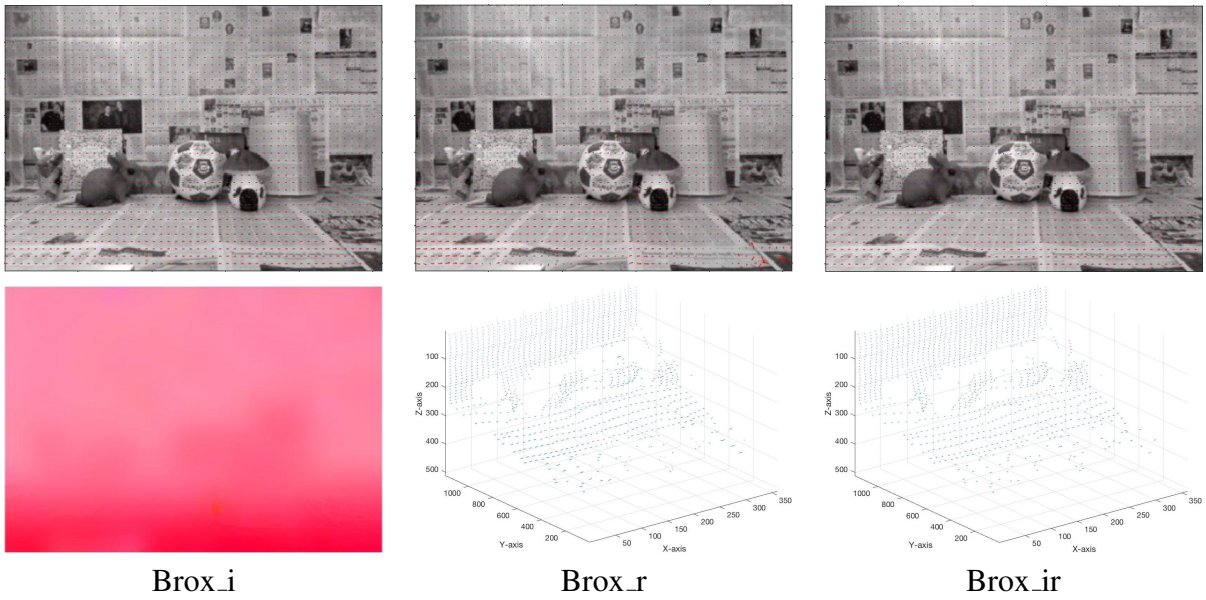


Figure 7.9: Brox Regularization flow using Kinect translation motion

7.9 3D Range Flow using R Data and IR Data Comparison

In this test, we demonstrated the effect when using only range data **(r)** to estimate 3D range flow compared to the combination of intensity and range data **(ir)**. In this experiment, we

executed all the 3D range approaches: **LS_r, TLS_r, Global_r, Global_ind_r, CLG_r, Brox_r, LS_ir, TLS_ir, Global_ir, GLobal_ind_ir, CLG_ir** and **Brox_ir**. Like the previous tests, this experiment was done without using any robust methods, except the **CLG** and **Brox** methods. In addition, we applied the **Kinect_trans** and **Kinect_div** datasets. We built a Gaussian pyramid that required four levels and $\eta = 0.5$. For Kinect_trans data, we chose frames 5 and 6, while in Kinect_div, we chose frames 3 and 4 for the test.

Table 7.8 shows the error measurements for the executed approaches in terms of the percentage of the flow density, AAE (Average Angular Error), EPE (End Point Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and the execution time in seconds.

This test shows that the 3D range/scene flow can be successfully estimated using only the depth data with **LS_r, TLS_r, Global_r, Global_ind_r, CLG_r** and **Brox_r** approaches. We were able to estimate the 3D range flow using the **local** and **global** approaches, in addition to **CLG** and **Brox** algorithms. Without any intensity data, we produced acceptable 3D range flow. These approaches can overcome barriers related to intensity problems when estimating the 3D scene flow because it can estimate flow in the dark (no need for light) with depth sensors instead of intensity cameras. However, when we applied the combination of intensity and depth data approaches, we significantly improved the 3D range flow in terms of **AAE**. Local approaches using combined data **(ir)** yield denser flow results compared with using depth only **(r)**.

Table 7.8: Error measurements to demonstrate the difference between R data and IR data

Kinect_Trans						
Method	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
LS_r	65.13	23.08	0.46	0.55	0.48	20.87
TLS_r	70.78	26.49	0.55	0.69	0.64	16.88
Global_r	100	17.01	0.45	0.63	1.10	4.15
Global_ind_r	100	21.36	0.46	0.56	0.62	34.68
CLG_r	100	11.09	0.26	0.32	0.50	14.34
Brox_r	100	11.40	0.27	0.31	0.53	15.33
LS_ir	98.73	18.59	0.62	0.75	2.00	13.00
TLS_ir	71.5	11.70	0.24	0.29	0.26	19.55
Global_ir	100	11.70	0.34	0.47	1.10	2.28
Global_ind_ir	100	16.00	0.39	0.53	0.72	26.74
CLG_ir	100	8.94	0.22	0.26	0.44	14.55
Brox_ir	100	8.95	0.22	0.27	0.45	13.98
Kinect_Div						
Method	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
LS_r	96.04	16.16	0.53	0.72	1.07	16.88
TLS_r	90.81	18.41	0.75	0.95	1.17	15.31
Global_r	100	18.28	0.63	0.90	1.22	7.93
Global_ind_r	100	7.06	0.24	0.35	0.87	25.06
CLG_r	100	18.15	0.49	0.68	0.96	16.36
Brox_r	100	17.65	0.52	0.65	1.28	13.53

LS_ir	98.72	17.29	0.73	0.95	1.75	19.16
TLS_ir	86.05	14.27	0.52	0.68	0.78	14.66
Global_ir	100	19.33	1.43	1.81	6.47	3.64
Global_ind_ir	100	7.25	0.25	0.38	0.87	24.58
CLG_ir	100	17.30	0.51	0.75	0.96	13.57
Brox_ir	100	18.52	0.55	0.78	1.13	13.42

7.10 Robust Effect

We tested several robust techniques to estimate 2D and 3D range flow. In Section 4.1.9, we explained the six robust techniques that can help produce a more robust optical flow. The six methods include: **Brox, Bruhn, Charbonnier, G-Charbonnier, Lorentzian** and **Geman-McClure**. In this experiment, we invoked the local and global approaches using three different data types (**i,r,ir**). We utilized **LS_i, LS_r, LS_ir, TLS_i, TLS_r, TLS_ir, Global_i, Global_r, Global_ir, Global_ind_i, Global_ind_r** and **Global_ind_ir**. Similarly,we used frames 5 and 6 from **Kinect_trans** datasets. The Gaussian pyramid was built with four levels and $\eta = 0.5$. Each robust method required a σ value, as shown in the formulations. In our test, we tried different σ values in each method; however, here we present only the best results.

Table 7.9 shows the error measurements using the percentage for flow density, AAE (Average Angular Error), RMSE (Root Mean Squared Error), and execution time in seconds.

In general, robust techniques did not have a significant impact on flow. Although they yielded a small improvement, they also required more time to execute. However, in local approaches with range data only, the robust methods did help to increase the density of the flow. In terms of AAE, there is only a small improvement in the flow direction. In each approach, a different robust method produced a better results. For example, in **LS_i**, **Bruhn** method gave the best improvement. Conversely, in **LS_r**, **Geman-McClure** gave the best flow. In **Global** approaches, **Lorentzian** improves the flow when we used a different value of the sigma for the data and smoothness term. **Bruhn** works well with the **Global_ind** approaches.

Table 7.9: Error measurements to demonstrate the robust techniques

Method	**Robust**	σ	**Density**	**AAE**	**RMSE**	**Time**
LS_i	Non	Non	100	8.53	0.43	26.87
	Brox	10	100	8.26	0.43	32.10
	Bruhn	0.001	100	8.16	0.43	61.95
	Charbonnier	10	100	8.31	0.43	30.67
	Generalized-Charbonnie	10	100	8.25	0.43	32.53
	Lorentzian	4	100	8.23	0.43	36.74
	Geman-McClure	5.5	100	8.23	0.43	58.86

. . . continued

Method	**Robust**	σ	**Density**	**AAE**	**RMSE**	**Time**
LS_r	Non	Non	65.13	23.08	0.48	20.87
	Brox	0.001	84.19	23.16	0.49	32.36
	Bruhn	0.001	65.13	23.08	0.48	25.25
	Charbonnier	0.001	99.23	32.52	2.00	32.19
	Generalized-Charbonnie	0.001	98.92	32.21	1.95	33.61
	Lorentzian	0.01	99.41	31.81	2.06	40.04
	Geman-McClure	0.01	78.94	22.91	0.48	27.50
LS_ir	Non	Non	98.73	18.59	2.00	13.00
	Brox	0.001	98.73	18.59	2.00	24.31
	Bruhn	0.001	98.73	18.59	2.00	17.41
	Charbonnier	0.001	98.73	18.59	2.00	25.10
	Generalized-Charbonnie	0.001	98.73	18.59	2.00	24.56
	Lorentzian	0.1	98.73	18.59	2.00	22.33
	Geman-McClure	0.01	98.73	18.59	2.00	15.53
TLS_i	Non	Non	100	8.48	0.44	35.30
	Brox	0.001	100	8.52	0.43	40.68
	Bruhn	0.001	100	8.48	0.44	46.83
	Charbonnier	0.001	100	8.47	0.43	42.00
	Generalized-Charbonnie	0.001	100	8.46	0.43	45.79
	Lorentzian	0.01	100	8.44	0.44	38.06
	Geman-McClure	0.01	100	8.48	0.44	44.25
TLS_r	Non	Non	70.78	26.49	0.64	16.88
	Brox	0.001	83.75	28.49	0.74	28.05
	Bruhn	0.001	71.43	26.34	0.64	23.82
	Charbonnier	0.001	83.97	28.83	0.76	25.66
	Generalized-Charbonnie	0.001	83.09	28.42	0.74	24.81
	Lorentzian	0.01	75.47	26.88	0.66	27.87
	Geman-McClure	0.01	73.41	26.46	0.64	24.08
TLS_ir	Non	Non	98.42	9.56	0.40	12.64
	Brox	0.001	98.79	9.73	0.80	17.10
	Bruhn	0.001	98.42	9.56	0.40	18.40
	Charbonnier	0.001	98.5	9.38	0.31	20.14
	Generalized-Charbonnie	0.001	98.77	9.71	0.75	16.36
	Lorentzian	0.01	98.48	9.57	0.41	16.37
	Geman-McClure	0.001	98.42	9.56	0.40	15.56
Global_i	Non	Non	100	8.75	0.44	7.57
	Brox	0.0001	100	8.55	0.44	25.36
	Bruhn	0.0001	100	8.55	0.44	24.59
	Charbonnier	0.001	100	8.58	0.43	28.47
	Generalized-Charbonnie	0.001	100	8.57	0.43	25.61

. . . continued

Method	**Robust**	σ	**Density**	**AAE**	**RMSE**	**Time**
	Lorentzian	6.5,0.001	100	8.55	0.43	6.70
	Geman-McClure	0.1	100	8.69	0.43	23.72
Global_r	Non	Non	100	17.01	1.10	4.15
	Brox	0.0001	100	16.22	1.60	18.75
	Bruhn	0.0001	100	16.22	1.60	18.65
	Charbonnier	0.0001	100	16.22	1.60	19.73
	Generalized-Charbonnie	0.0001	100	16.08	1.69	19.07
	Lorentzian	0.0001	100	15.70	1.70	12.06
	Geman-McClure	0.001	100	15.70	1.64	13.40
Global_ir	Non	Non	100	11.70	1.10	2.28
	Brox	0.0001	100	10.88	0.50	14.73
	Bruhn	0.0001	100	14.20	10.17	11.76
	Charbonnier	0.0001	100	10.85	0.50	17.49
	Generalized-Charbonnie	0.0001	100	10.86	0.50	13.94
	Lorentzian	0.0001	100	10.78	0.52	22.41
	Geman-McClure	10	100	13.69	0.52	14.74
Global_in_i	Non	Non	100	8.52	0.43	25.72
	Brox	10	100	8.46	0.43	14.70
	Bruhn	0.001	100	8.48	0.43	24.20
	Charbonnier	1,0.0001	100	8.47	0.43	17.32
	Generalized-Charbonnie	1,0.0001	100	8.47	0.43	15.29
	Lorentzian	1,0.0001	100	8.47	0.43	17.03
	Geman-McClure	1, 0.0001	100	8.47	0.43	17.64
Global_in_r	Non	Non	100	21.36	0.62	34.68
	Brox	1	100	21.94	0.63	19.92
	Bruhn	0.00001	100	21.12	0.61	28.51
	Charbonnier	1	100	22.01	0.64	25.76
	Generalized-Charbonnie	0.00001	100	21.12	0.61	28.51
	Lorentzian	10, 0.03	100	21.73	0.63	17.09
	Geman-McClure	6.5,0.03	100	21.73	0.63	16.95
Global_in_ir	Non	Non	100	14.64	0.60	25.50
	Brox	1,0.0001	100	13.07	0.53	24.90
	Bruhn	0.0001	100	13.06	0.53	28.37
	Charbonnier	1,0.0001	100	13.07	0.53	25.41
	Generalized-Charbonnie	1,0.0001	100	13.07	0.53	25.97
	Lorentzian	10, 0.03	100	14.31	0.57	26.93
	Geman-McClure	1, 0.0001	100	13.73	1.48	25.30

7.11 Weighted Intensity and Range Differentiation Effect

In this test, we demonstrated the effect of weighting intensity and range derivatives. As described in Section 4.1.4, we needed to use a factor value β to weight the intensity and depth derivatives with approaches that combined intensity and range data. Therefore, this experiment used **LS_ir, TLS_ir, Global_ir, Global_ind_ir, CLG_ir**, and **Brox_ir** approaches with **Kinect_trans** datasets using frames 5 and 6. We utilized four levels with $\eta = 0.5$ to build the Gaussian pyramid.

Table 7.10 shows the two options for β values that we tested with each approach. Using Equation 4.1, $\beta = 45.8$ when using the derivatives of frames 5 and 6 in the **Kinect_trans** datasets. Additionally, we tested $\beta = 1$ to demonstrate the absent effect of weighting. We show the error measurements in terms of the percentage of flow density, AAE (Average Angular Error), EPE (End Point Error), MAS (Mean Absolute Error), RMSE (Root Mean Squared Error), and execution time in seconds.

Using the weighted $\beta = 45.8$, we obtained better flow using all the invoked approaches that we tried. In local methods (**LS_ir, TLS_ir**), the density increased, and the AAE improved noticeably. In the regularization methods, we obtained a 100% flow in both cases. Notably, the AAE is better when $\beta = 45.8$.

Table 7.10: Error measurements to demonstrate the effect of weighted the intensity and range derivatives

Method	β	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
LS_ir	1	97.75	19.25	0.77	0.92	3.09	13.07
	45.8	98.73	18.59	0.62	0.75	2.00	13.00
TLS_ir	1	71.5	11.70	0.24	0.29	0.26	19.55
	45.8	98.42	9.56	0.22	0.26	0.40	12.64
Global_ir	1	100	15.97	0.42	0.56	0.94	3.24
	45.8	100	11.70	0.34	0.47	1.10	2.28
Global_ind_ir	1	100	16.00	0.39	0.53	0.72	26.74
	45.8	100	14.64	0.35	0.46	0.60	25.50
CLG_ir	1	100	8.94	0.22	0.26	0.44	14.55
	45.8	100	8.91	0.22	0.26	0.47	22.85
Brox_ir	1	100	9.70	0.24	0.28	0.49	21.43
	45.8	100	8.95	0.22	0.27	0.45	13.98

7.12 Weighted Median Filter Effect

In Section 4.1.7, we described the weighted median filter used in our model. We applied this filter for the intermediate flow to reduce the outliers and smooth the output flow. In this

test, we demonstrated the effect of this weighted median filter. This time, we invoked **Global** approaches with the three different data types (**i, r, ir**) (the other approaches react similarly to the weighted median filter). We used **Kinect_trans** datasets, and we again used frames 5 and 6. Like our previous tests, we used four levels and $\eta = 0.5$ to build the Gaussian pyramid. This test was also done without any robust techniques.

Table 7.11 shows the two cases in which median filter was applied (**Y,N**). **Y** demonstrates the application of the filter and **N** demonstrates the application without the filter. The error measurements are shown in terms of the percentage of the flow density, AAE (Average Angular Error), EPE (End Point Erro), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and execution time in seconds.

When applying the weighted median filter, the execution time increased because of the extra steps required; however, the error measurements decreased to produce a better output flow. Using either intensity only data (**i**) or the range only data (**r**), this improvement is good even if the differences are small. However, using the combined approach (**ir**), the improvement was significant.

Table 7.11: Error measurements to demonstrate the effect of weighted median filter

Method	Filter?	Density	AAE	EPE	MAE	RMSE	Time
Global_i	Y	100	8.75	0.22	0.24	0.44	7.57
	N	100	9.52	0.23	0.27	0.45	1.46
Global_r	Y	100	17.01	0.45	0.63	1.10	4.15
	N	100	17.49	0.48	0.68	1.25	2.56
Global_ir	Y	100	11.70	0.34	0.47	1.10	2.28
	N	100	19.14	0.50	0.70	1.06	1.32

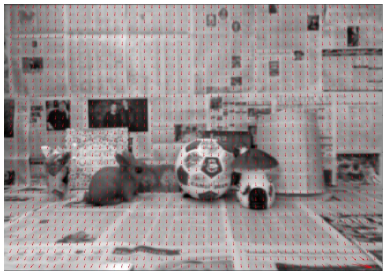
7.13 Translation and Divergence Motions

As mentioned in Chapter 6, the data gained from the Kinect V2 camera was captured in two different motions: translation in the x-direction and diverging toward the scene in the z-direction. In the previous sections, we used **Kinect_trans** datasets, which have translation motion in the x-direction. In this next experiment, we demonstrated the effect of the implemented approaches with the divergence motion. Specifically, we used the **Kinect_div** datasets, which have divergence motion in the z-direction. When testing all the 18 approaches, we used four levels and $\eta = 0.5$ to build the Gaussian pyramid. Notably, divergence motion is more complicated than translation motion. Specifically, it has two different direction motions at the same time: (u,v) move in a divergence direction, while the value of (w) is changed by either increasing or decreasing the number of depth values.

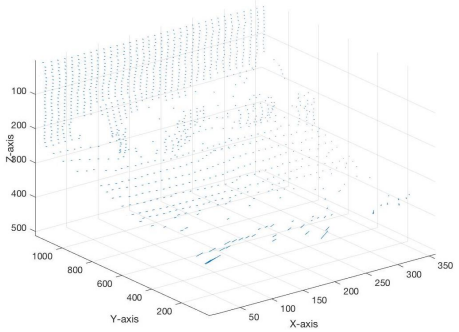
We choose frames 3 and 4 for this experiment. The best estimation of the correct flow (ground truth flow) using frames 3 and 4 is shown in Figure 7.10 as a vector representation of (u,v) and

(u,w) plus the 3D plot of (u,v,w).

Vector Flow for (u,v)



Vector Flow for (u,w)



3D Flow

Figure 7.10: Correct flow for the Kinect divergence motion using frame 3 and 4

Local Approaches:
We used local approaches (**LS, TLS**) to estimate the Kinect divergence motion. For the **LS** approaches, we need one threshold value; however, the **TLS** approaches require two threshold values for 2D optical flow and three thresholds values for the 3D range flow. In this experiment, we used $\tau = 10$ with **LS_i**, $\tau = 0.3$ with **LS_r** and $\tau = 0.1$ in **LS_ir**. For **TLS_i** we used $\tau 1 = 20, \tau 2 = 20$, **TLS_r** utilized $\tau 1 = 30, \tau 2 = 30, \tau 3 = 1$. Lastly, **TLS_ir** used $\tau 1 = 20, \tau 2 = 20, \tau 3 = 0.1$. The error measurements for the local approaches using these thresholds are shown in Table 7.12.

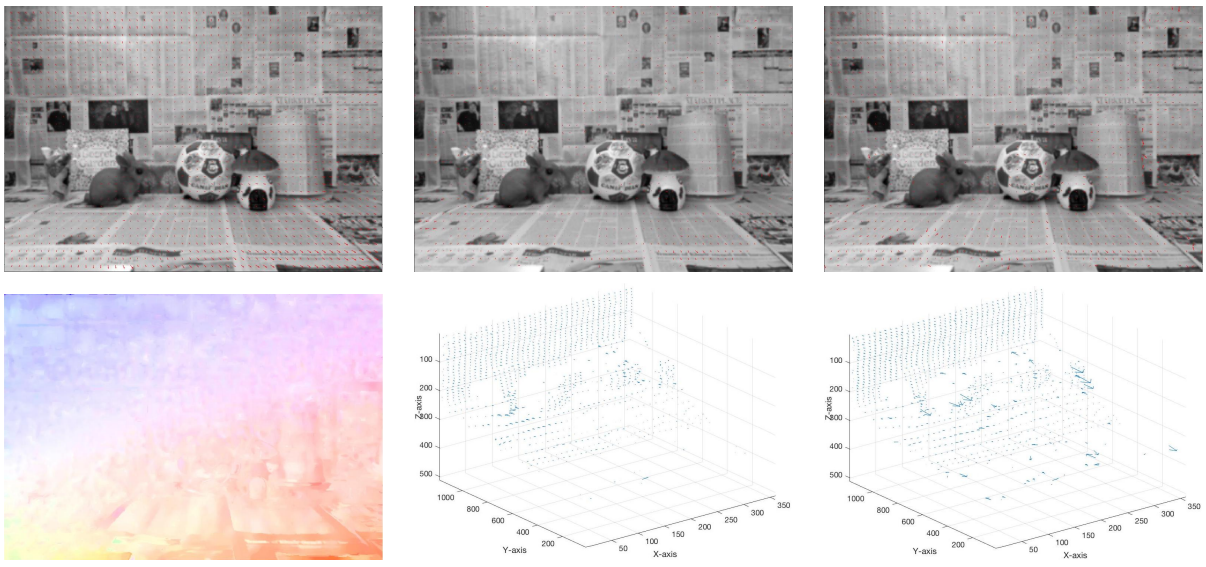
Table 7.12 shows the error measurements in terms of the percentage of the flow density, AAE (Average Angular Error), EPE (End Point Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and execution time in seconds.

Using **LS_i** and **TLS_i**, we can estimate close 2D optical flow. In both approaches, we can obtain 100% dense flow. In 3D range flow, **LS** estimates produce a slightly better 3D flow than **TLS** approach. Notably, it is clear that the combined approaches **LS_ir** and **TLS_ir** provide better 3D flow than either **LS_r**, **TLS_r** approaches.

Figure 7.11 shows the flow estimated using the Least Square approaches **(LS)**. 2D optical flow is shown in a 2D vector and colour representation. 3D range flow is shown as a 2D vector and in a 3D plot for (u,v,w) flow. Figure 7.12 shows the estimated flow using Total Least Square methods **(TLS)**.

Table 7.12: Error measurements for LS and TLS approaches

Method	Density	AAE	EPE	MAE	RMSE	Time
LS_i	100	19.43	0.41	0.52	0.91	21.47
LS_r	96.04	16.16	0.53	0.72	1.07	16.88
LS_ir	98.72	17.29	0.73	0.95	1.75	19.16
TLS_i	100	21.80	0.47	0.58	0.95	22.4
TLS_r	90.81	18.41	0.75	0.95	1.17	15.31
TLS_ir	86.05	14.27	0.52	0.68	0.78	14.66



LS_i LS_r LS_ir

Figure 7.11: Least Square flow using Kinect divergence motion

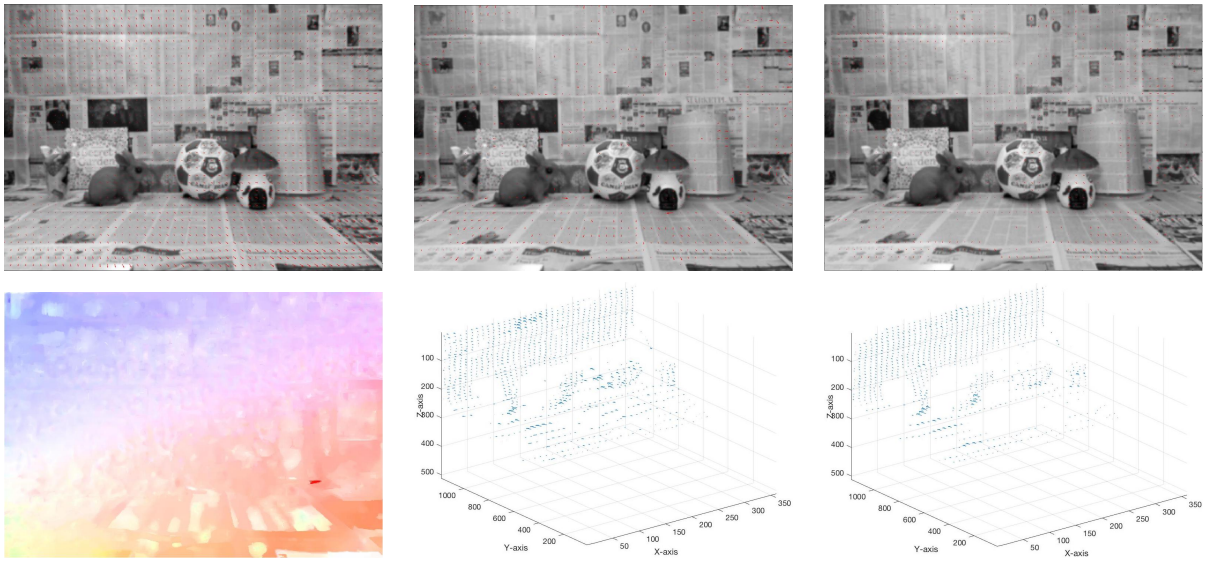


Figure 7.12: Total Least Square flow using Kinect divergence motion

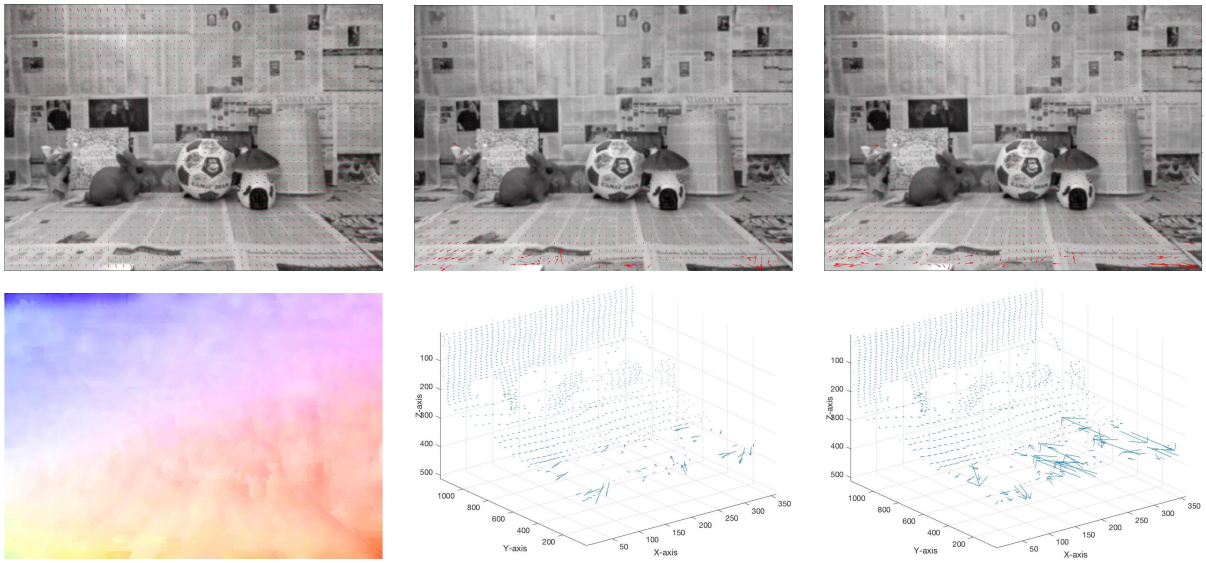
Global Approaches:
Table 7.13 presents the error measurements to demonstrate the difference between direct and indirect global approaches. This table shows the percentage of the density for the output flow, AAE (Average Angular Error), EPE (End Point Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and execution time in seconds.

Using direct and indirect global approaches, we can estimate 100% dense flow in both 2D optical flow and 3D range flow. The AAE and RMSE are better in the indirect global approach. Overall, indirect approaches need more time because of the local estimation step, but they do produce good flow. The output flow using indirect approaches are more robust and do not have outliers like **Global_r** approach. However, estimating the 3D range flow using the combination of intensity and range data improves the output flow than using depth information only.

Figure 7.13 shows the flow estimated using the direct global approaches **Global**. 2D optical flow is shown in a 2D vector and color representation. 3D range flow is shown as a 2D vector and in a 3D plot for (u,v,w) flow. Figure 7.14 shows the flow estimated using Indirect Global methods **Global_ind**.

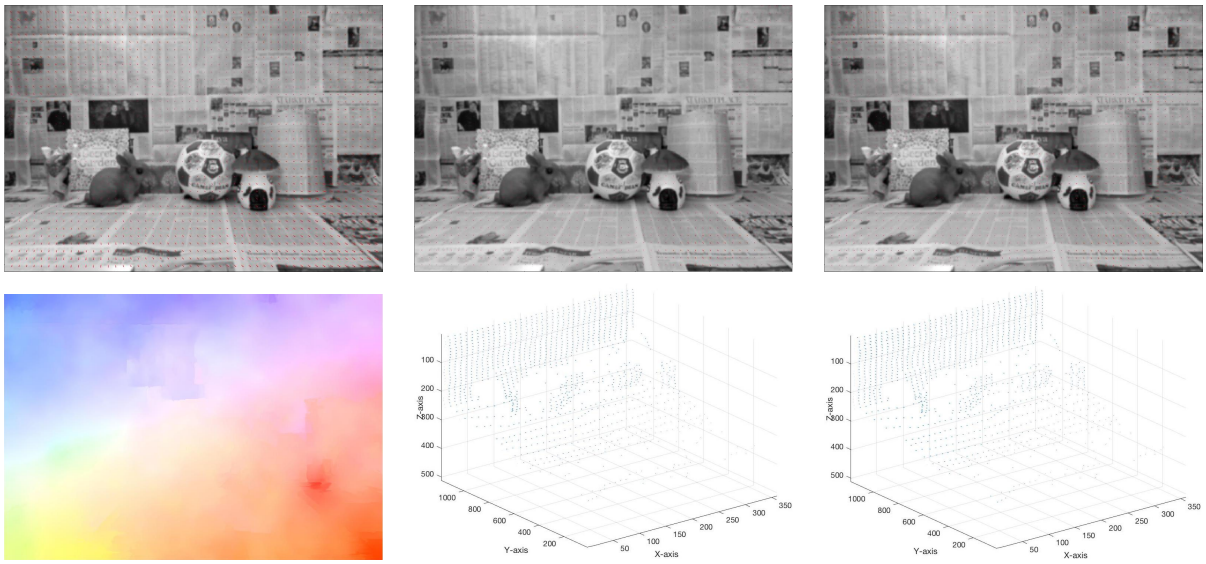
Table 7.13: Error measurements for Direct Global and Indirect Global approaches

Method	Density	AAE	EPE	MAE	RMSE	Time
Global_i	100	18.93	0.41	0.51	0.91	5.86
Global_r	100	18.28	0.63	0.90	1.22	7.93
Global_ir	100	19.33	1.43	1.81	6.47	3.64
Global_ind_i	100	17.95	0.38	0.47	0.89	29.21
Global_ind_r	100	7.06	0.24	0.35	0.87	25.06
Global_ind_ir	100	7.25	0.25	0.38	0.87	24.58



Global_i Global_r Global_ir

Figure 7.13: Direct Regularization flow using Kinect divergence motion



Global_ind_i Global_ind_r Global_ind_ir

Figure 7.14: Indirect Regularization flow using Kinect divergence motion

CLG and Brox Approaches:
Using Bruhn et al.'s method (**CLG**) and Brox et al.'s method (**Brox**), the 2D optical flow and 3D range flow can be estimated using a divergence motion. To demonstrate the effect of

using these methods, we estimated 2D optical flow and 3D range flow using the intensity data **(i)**, range data **(r)**, and the combination of intensity and range data **(ir)**. For this experiment, we executed **CLG_i, CLG_r, CLG_ir, Brox_i, Brox_r** and **Brox_ir** approaches. In the CLG method, Bruhn's robust methods were used as described in their algorithm; similarly, we also applied the Brox robust techniques as described in their algorithm.

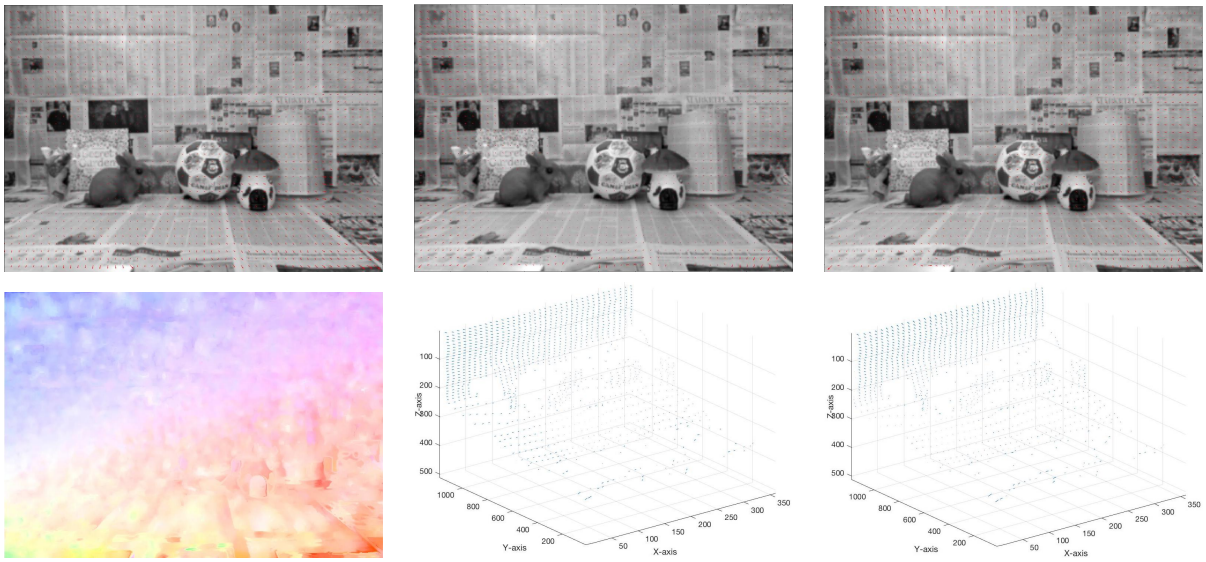
Table 7.14 shows the error measurements for the CLG and Brox approaches in term of the percentage of the output flow, AAE (Average Angular Error), EPE (End Point Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and execution time in seconds.

Using CLG and Brox methods, we can obtain 100% dense of the flow in both 2D optical and 3D range flow. The AAE and RMSE for both methods are very similar. However, the visualization figures for the output flow of the CLG and Brox indicate that the Brox methods can give a smoother and denser optical flow (even if the measurements did not explictly show that).

Figure 7.15 shows the divergence flow estimated using the Bruhn et al. method **CLG**. 2D optical flow is shown in a 2D vector and color representation. 3D range flow is shown as a 2D vector and in a 3D plot for (u,v,w) flow. Figure 7.16 shows the flow estimated using Brox et al.'s methods **Brox**.

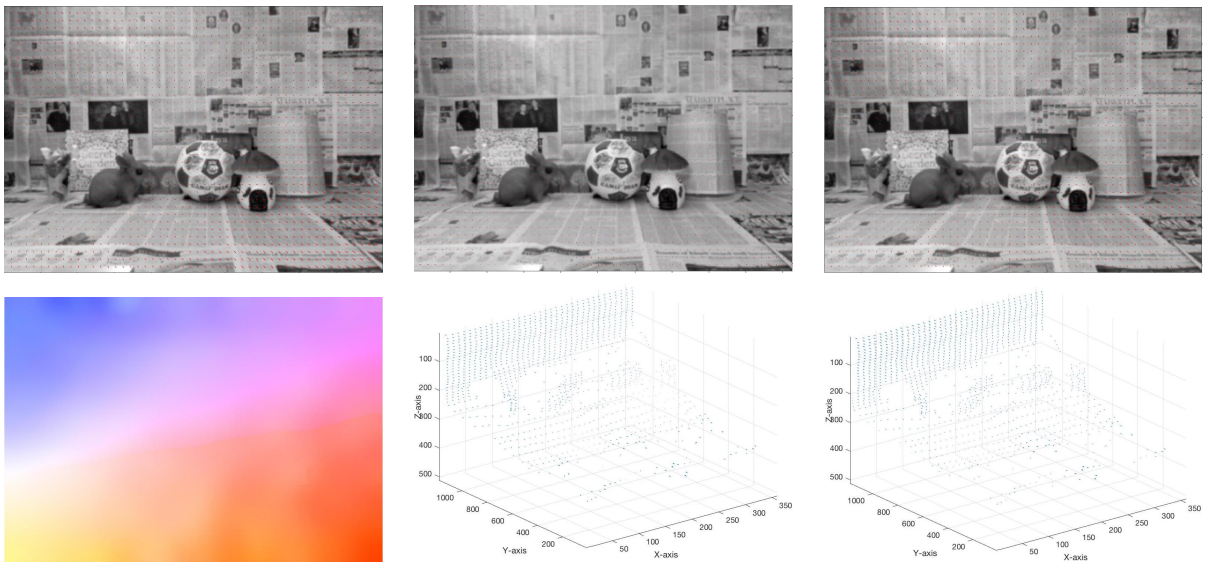
Table 7.14: Error measurements for Direct Global and Indirect Global approaches

Method	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
CLG_i	100	16.08	0.34	0.43	0.87	23.01
GLG_r	100	18.15	0.49	0.68	0.96	16.36
CLG_ir	100	17.30	0.51	0.75	0.96	13.57
Brox_i	100	19.05	0.40	0.51	0.90	12.24
Brox_r	100	17.65	0.52	0.65	1.28	13.53
Brox_ir	100	18.52	0.55	0.78	1.13	13.42



CLG_i CLG_r CLG_ir

Figure 7.15: CLG Regularization flow using Kinect divergence motion



Brox_i Brox_r Brox_ir

Figure 7.16: Brox Regularization flow using Kinect divergence motion

7.14 Computation Time

The execution time of the algorithms is presented in Table 7.15. The algorithms were implemented and tested using MATLAB_2018a in a Mac machine running the macOS High Sierra (10.13.6) operating system. The processor was 3 GHz Intel Core i7 and 16 GB 1600 MHz

DDR3 memory. This machine does not have GPU; therefore, the CPU time for each approach is presented in seconds. In this test, we show the computation time for both translation and divergence motions using Kinect V2 camera when using all 18 approaches. Recall, the data has size 360×512. MATLAB usually gives slightly different times in each run; therefore, we have computed the average of three execution times for each approach. Generally, this time in seconds is not slow because we are vectorizing our code in most cases to avoid the loops and iteration computation. The maximum time was about 30 seconds, which was for the indirect global approaches that require the least square computation in the first step. Consequently, the maximum time to estimate 2D/3D flow using our approaches is 30 seconds/frame.

Table 7.15: Computation Time for the Kinect datasets

Method	**Kinect_trans**	**Kinect_Div**
LS_i	26.87	21.47
LS_r	20.87	16.88
LS_ir	13.00	19.16
TLS_i	35.30	22.4
TLS_r	16.88	15.31
TLS_ir	19.55	14.66
Global_i	7.57	5.86
Global_r	4.15	7.93
Global_ir	2.28	3.64
Global_ind_i	25.72	29.21
Global_ind_r	34.68	25.06
Global_ind_ir	26.74	24.58
CLG_i	13.37	23.01
CLG_r	14.34	16.36
CLG_ir	14.55	13.57
Brox_i	12.24	12.24
Brox_r	15.33	13.53
Brox_ir	13.98	13.42

7.15 Camera Effect

As mentioned in Chapter 4, we generated optical/range datasets using different cameras. Specifically, we used Kinect V2, ZED, Truedepth (Front iPhone X), and the rear camera in the iPhone X to capture the same scene. Importantly, we could not gain a 100% identical scene from these cameras because each camera has a different field of view (FOV) and focal length (f), even if we captured the images from the same position. Therefore, we show 2D/3D optical and range flow using **CLG** and **Brox** approaches by utilizing the three different datatypes: by using intensity only **(i)**, or using range data only **(r)** or combining intensity with range data together **(ir)**. We also tested the translation and divergence motions for each camera.

7.15.1 Kinect Datasets

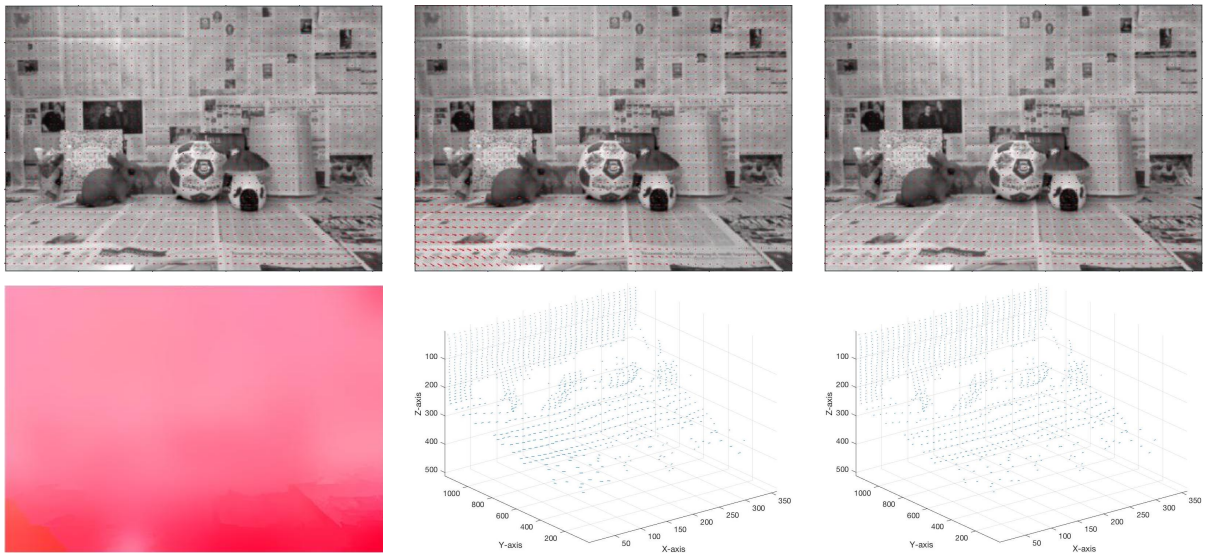
As described throughout this chapter, we have already tested the Kinect datasets in several experiments. In this section, we recap the results for **CLG** and **Brox** approaches using the translation and divergence motions. In this test, we used frames 5 and 6 in (**Kinect_trans**) datasets, while we used frames 3 and 5 in (**Kinect_div**). We utilized four levels and $\eta = 0.5$ to build the Gaussian pyramid.

Table 7.16 shows the error measurements in terms of the flow density, AAE (Average Angular Error), EPE (End Point Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and execution time in seconds.

Figure 7.17 and Figure 7.18 show the translation motion using **CLG** and **Brox** approaches, while Figure 7.19 and Figure 7.20 represent the divergence motion using **CLG** and **Brox** approaches, respectively.

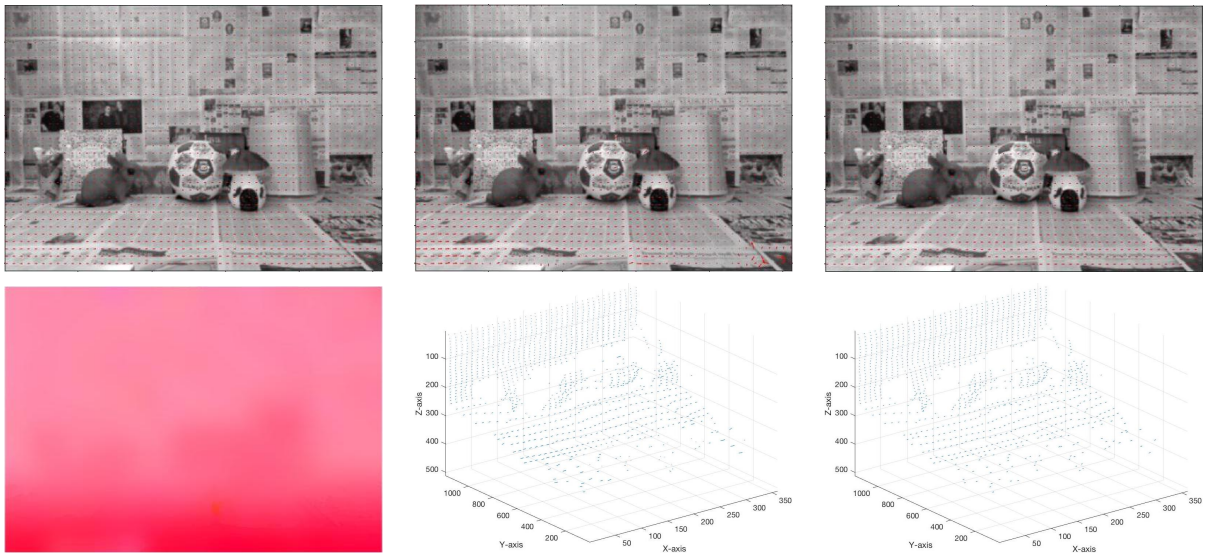
Table 7.16: Error measurements for Kinect datasets

Kinect_Trans						
Method	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
CLG_i	100	8.53	0.22	0.23	0.44	13.37
CLG_r	100	11.09	0.26	0.32	0.50	14.34
CLG_ir	100	8.94	0.22	0.26	0.44	14.55
Brox_i	100	8.47	0.21	0.23	0.44	12.24
Brox_r	100	11.40	0.27	0.31	0.53	15.33
Brox_ir	100	8.95	0.22	0.27	0.45	13.98
Kinect_Div						
Method	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
CLG_i	100	16.08	0.34	0.43	0.87	23.01
GLG_r	100	18.15	0.49	0.68	0.96	16.36
CLG_ir	100	17.30	0.51	0.75	0.96	13.57
Brox_i	100	19.05	0.40	0.51	0.90	12.24
Brox_r	100	17.65	0.52	0.65	1.28	13.53
Brox_ir	100	18.52	0.55	0.78	1.13	13.42



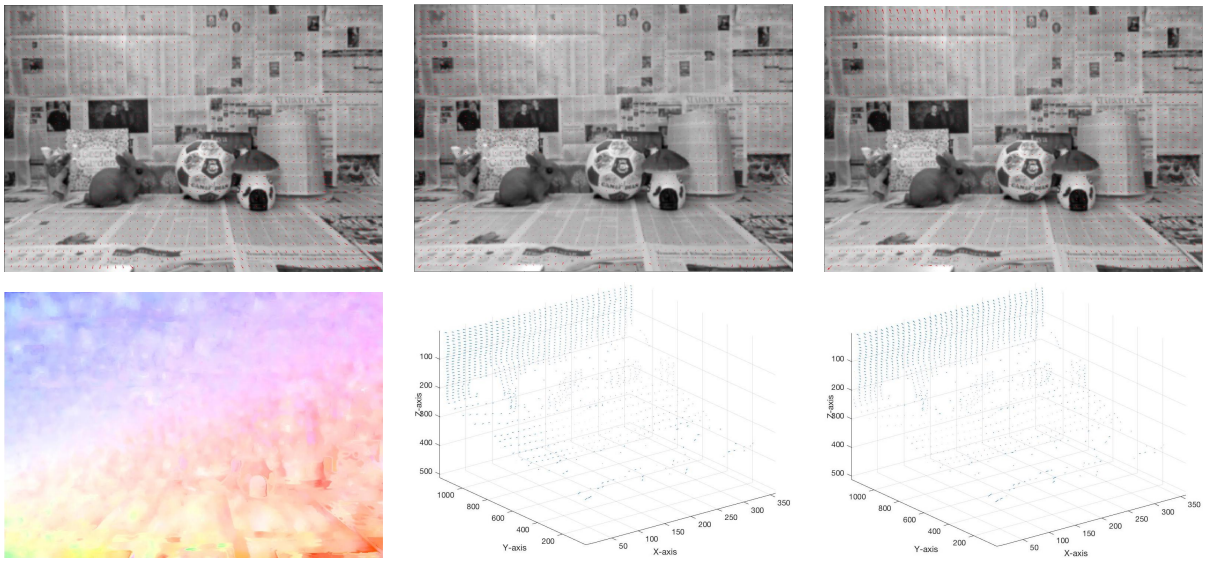
CLG_i CLG_r CLG_ir

Figure 7.17: CLG Regularization flow using Kinect translation motion



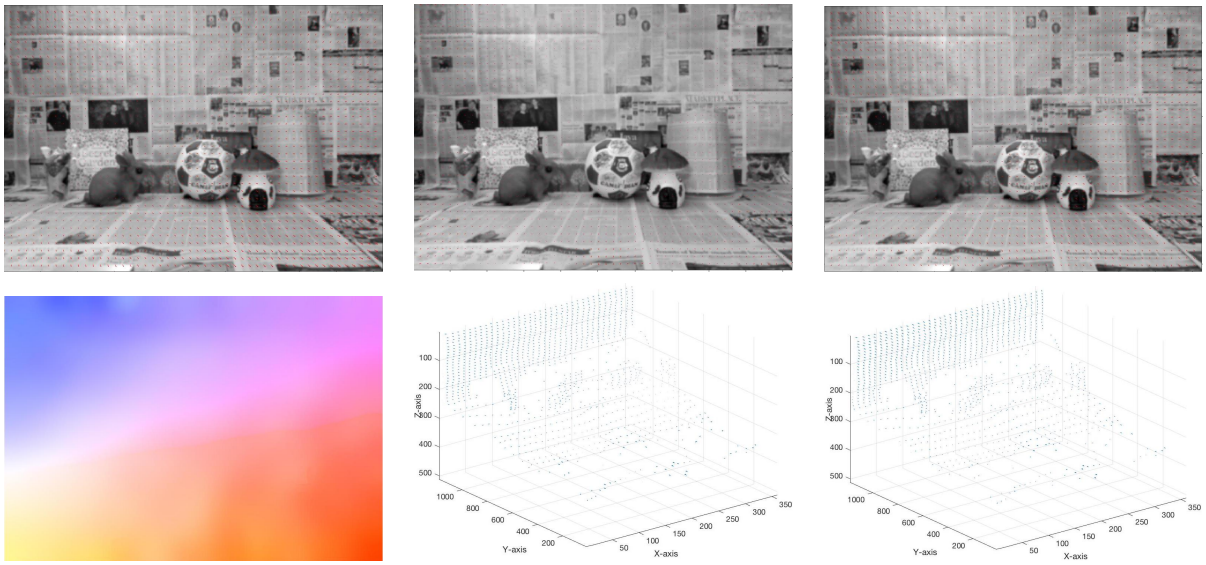
Brox_i Brox_r Brox_ir

Figure 7.18: Brox Regularization flow using Kinect translation motion



CLG_i CLG_r CLG_ir

Figure 7.19: CLG Regularization flow using Kinect divergence motion



Brox_i Brox_r Brox_ir

Figure 7.20: Brox Regularization flow using Kinect divergence motion

7.15.2 ZED Datasets

Using the ZED camera, we generated two different datasets. **ZED_trans** and **ZED_div** for translation and divergence motions respectively. We estimated 2D/3D flow using **CLG** and **Brox** approaches with the three different datasets (**i,r,ir**). In this test, we used half the size of the ZED intensity frames (grayvalue images) and depth frames to reduce the execution time.

The new size was 540×960. In this test, we used frames 1 and 2 in **ZED_trans** datasets, while we used frames 3 and 4 in **ZED_div**. We utilized *max* levels and $\eta = 0.95$ to build Brox pyramid.

Table 7.17 shows the error measurements in terms of the flow density, AAE (Average Angular Error), EPE (End Point Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and execution time in seconds.

Figure 7.21 and Figure 7.22 show the translation motion using **CLG** and **Brox** approaches, while Figure 7.23 and Figure 7.24 represent the divergence motion using **CLG** and **Brox** approaches, respectively.

Table 7.17: Error measurements for ZED datasets

ZED_Trans						
Method	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
CLG_i	100	11.13	0.32	0.35	0.33	122.30
CLG_r	100	14.00	0.37	0.52	0.39	220.12
CLG_ir	100	12.04	0.33	0.44	0.34	218.22
Brox_i	100	11.12	0.32	0.33	0.33	453.93
Brox_r	100	16.25	0.43	0.59	0.48	538.62
Brox_ir	100	14.82	0.39	0.53	0.42	522.09
ZED_Div						
Method	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
CLG_i	100	5.15	0.10	0.13	0.14	461.24
CLG_r	100	17.52	0.54	0.82	0.60	139.36
CLG_ir	100	7.65	0.26	0.40	0.33	233.83
Brox_i	100	4.37	0.08	0.11	0.09	400.86
Brox_r	100	15.59	0.75	0.95	1.04	498.82
Brox_ir	100	6.92	0.28	0.37	0.29	264.67

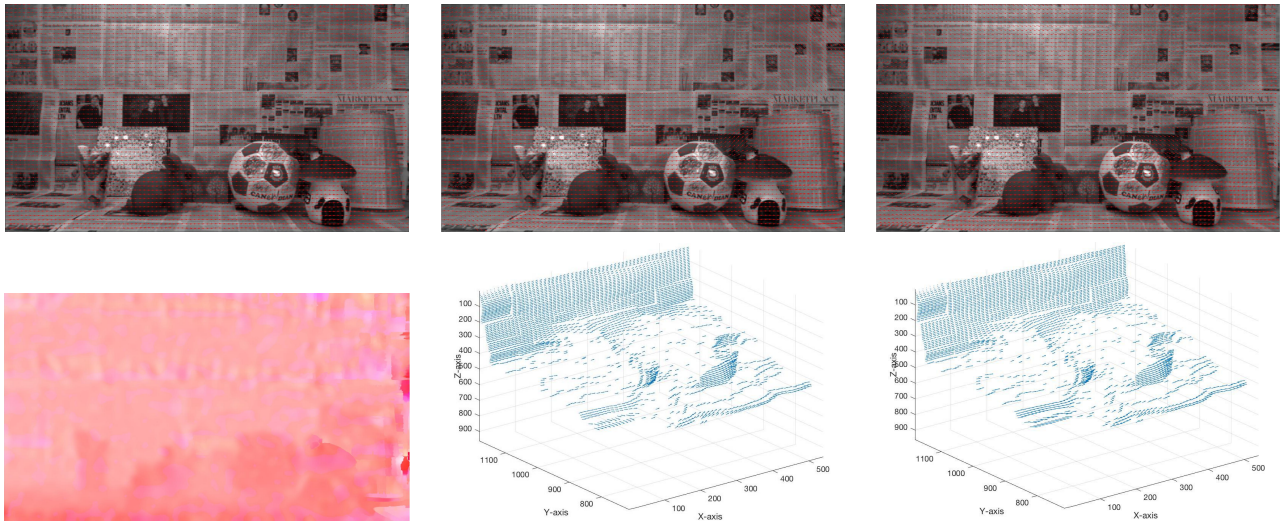


Figure 7.21: CLG Regularization flow using ZED translation motion

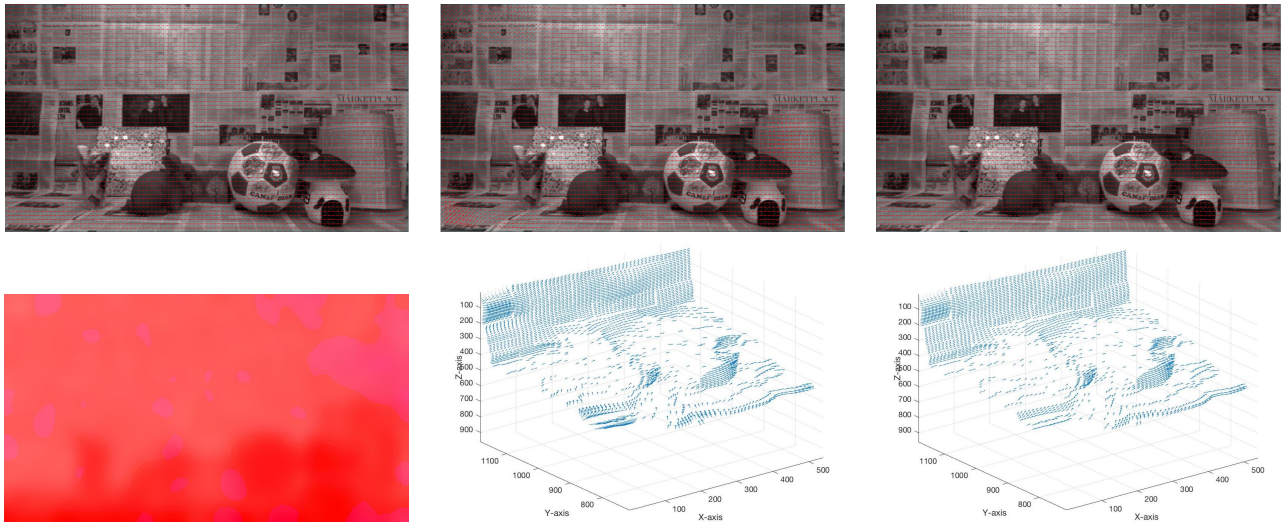


Figure 7.22: Brox Regularization flow using ZED translation motion

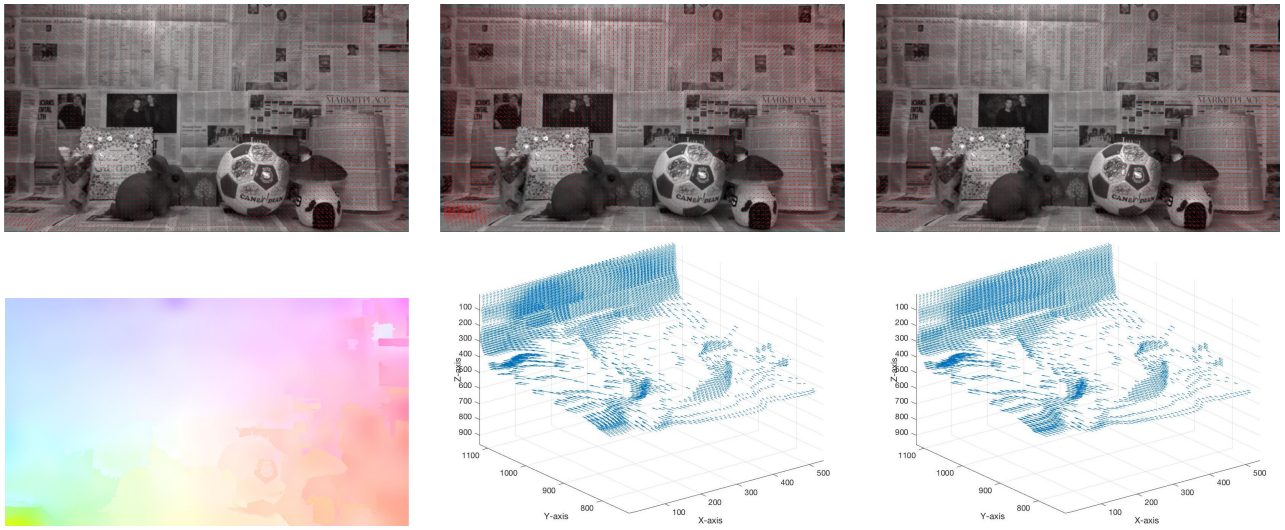


Figure 7.23: CLG Regularization flow using ZED divergence motion

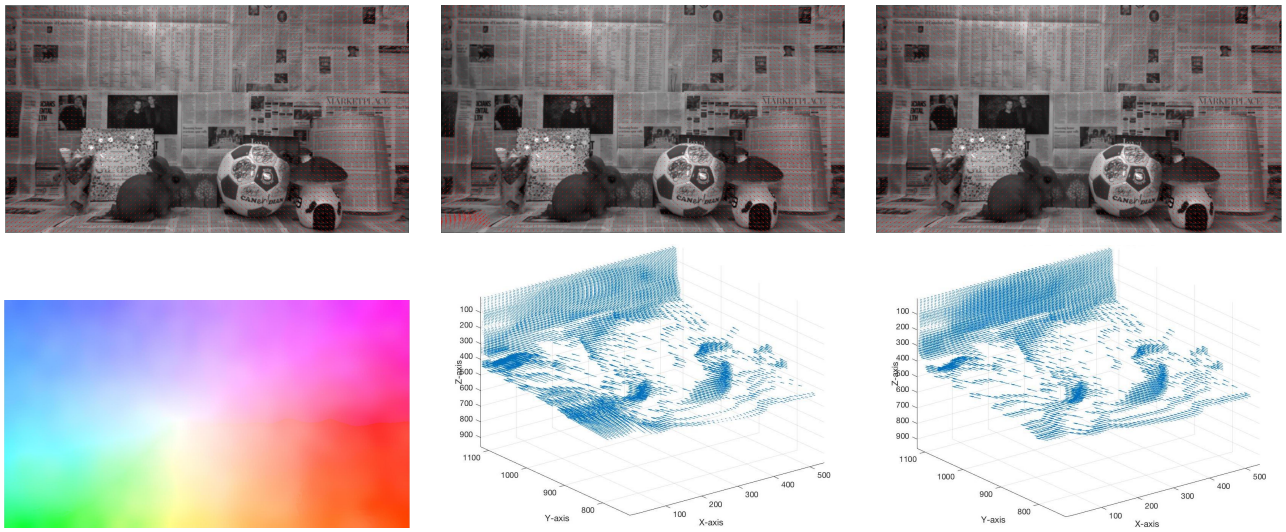


Figure 7.24: Brox Regularization flow using ZED divergence motion

7.15.3 Front iPhoneX Camera (Truedepth) Datasets

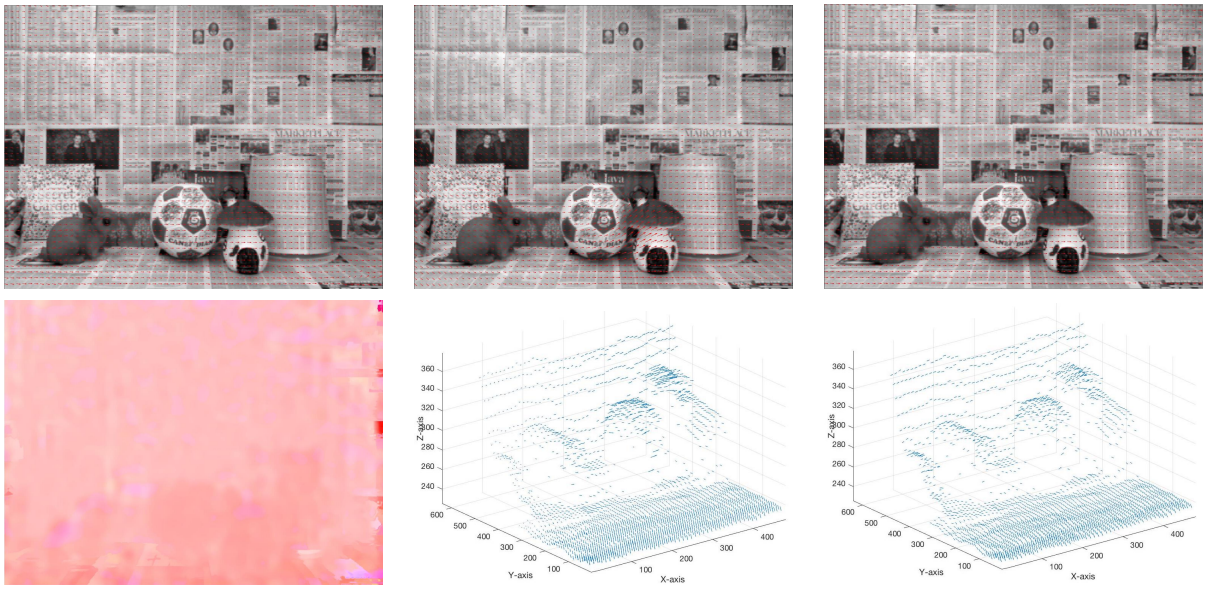
Using the Truedepth camera in iPhone X, we generated two different datasets: **front_trans** and **front_div** for translation and divergence motions, respectively. We estimated 2D/3D flow using **CLG** and **Brox** approaches with the three different datasets (**i,r,ir**). We are scaling down the colour images to had same size like depth images. The image sequences using front camera had size 480×640. In this test, we used frames 1 and 2 in **front_trans** datasets, while we used frames 3 and 4 in **front_div**. We utilized *max* levels and $\eta = 0.95$ to build Brox pyramid.

Table 7.18 shows the error measurements in terms of the flow density, AAE (Average Angular Error), EPE (End Point Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and execution time in seconds.

Figure 7.25 and Figure 7.26 show the translation motion using **CLG** and **Brox** approaches, while Figure 7.27 and Figure 7.28 represent the divergence motion using **CLG** and **Brox** approaches, respectively.

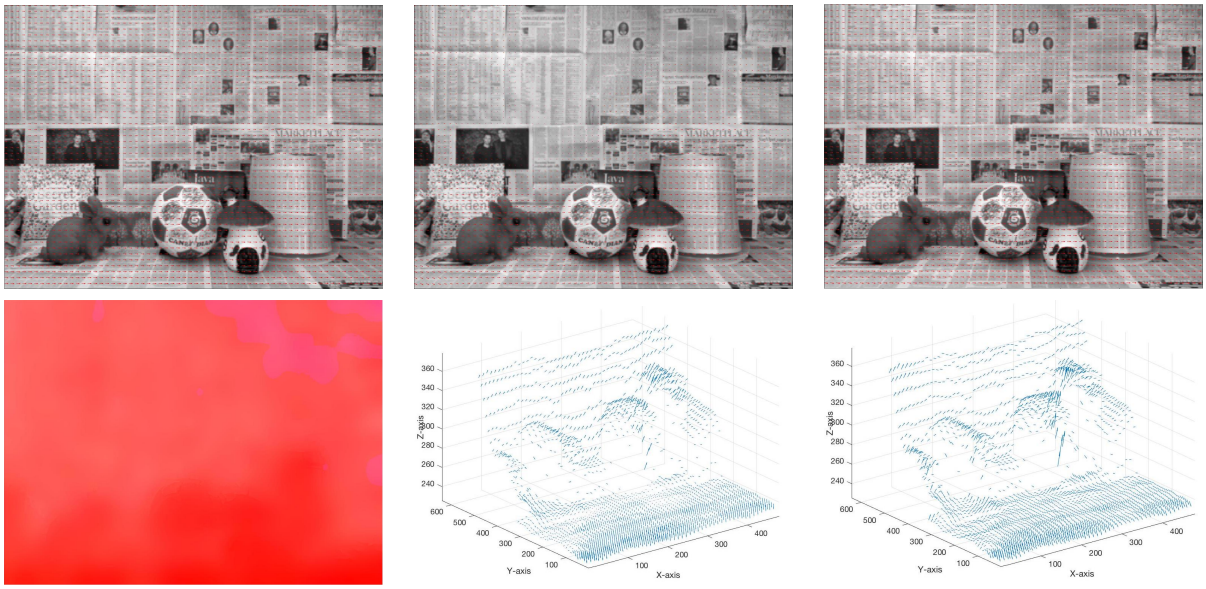
Table 7.18: Error measurements for Truedepth (front) camera in iPhone X datasets

Front_Trans						
Method	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
CLG_i	100	22.09	0.65	0.68	0.68	54.85
CLG_r	100	30.12	0.79	1.07	0.85	64.25
CLG_ir	100	24.58	0.68	0.88	0.73	104.72
Brox_i	100	22.14	0.65	0.67	0.68	178.20
Brox_r	100	35.66	0.91	1.14	0.96	172.36
Brox_ir	100	25.47	0.70	0.90	0.74	244.06
Front_Div						
Method	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
CLG_i	100	26.10	0.61	0.79	0.67	62.93
CLG_r	100	53.19	1.33	2.06	1.34	78.30
CLG_ir	100	47.72	1.23	1.83	1.24	118.50
Brox_i	100	26.18	0.61	0.79	0.67	169.99
Brox_r	100	59.84	1.45	2.14	1.47	244.18
Brox_ir	100	57.33	1.41	2.04	1.42	258.87



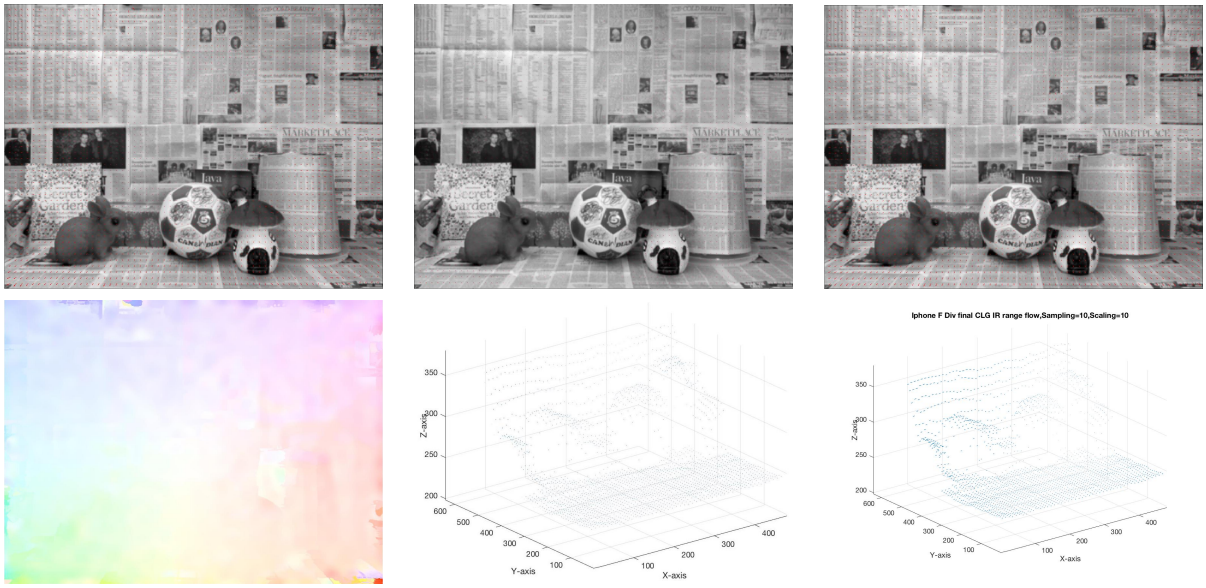
CLG_i CLG_r CLG_ir

Figure 7.25: CLG Regularization flow using Truedepth camera in a translation motion



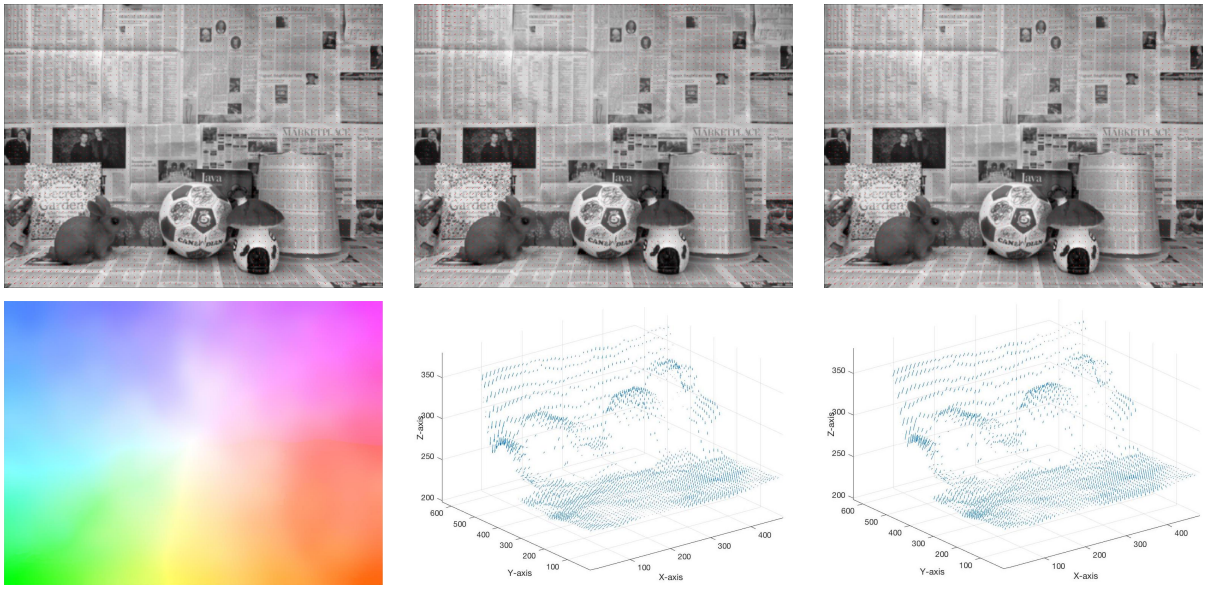
Brox_i Brox_r Brox_ir

Figure 7.26: Brox Regularization flow using Truedepth camera in translation motion



CLG_i CLG_r CLG_ir

Figure 7.27: CLG Regularization flow using Truedepth camera in divergence motion



Brox_i Brox_r Brox_ir

Figure 7.28: Brox Regularization flow using Truedepth camera in divergence motion

7.15.4 Rear iPhoneX Camera (Stereo Vision) Datasets

Using the rear camera in iPhpne X, we generated two different datasets: **rear_trans** and **rear_div** for translation and divergence motions, respectively. We estimated 2D/3D flow using **CLG** and **Brox** approaches with the three different datasets (**i,r,ir**). The image sequences using

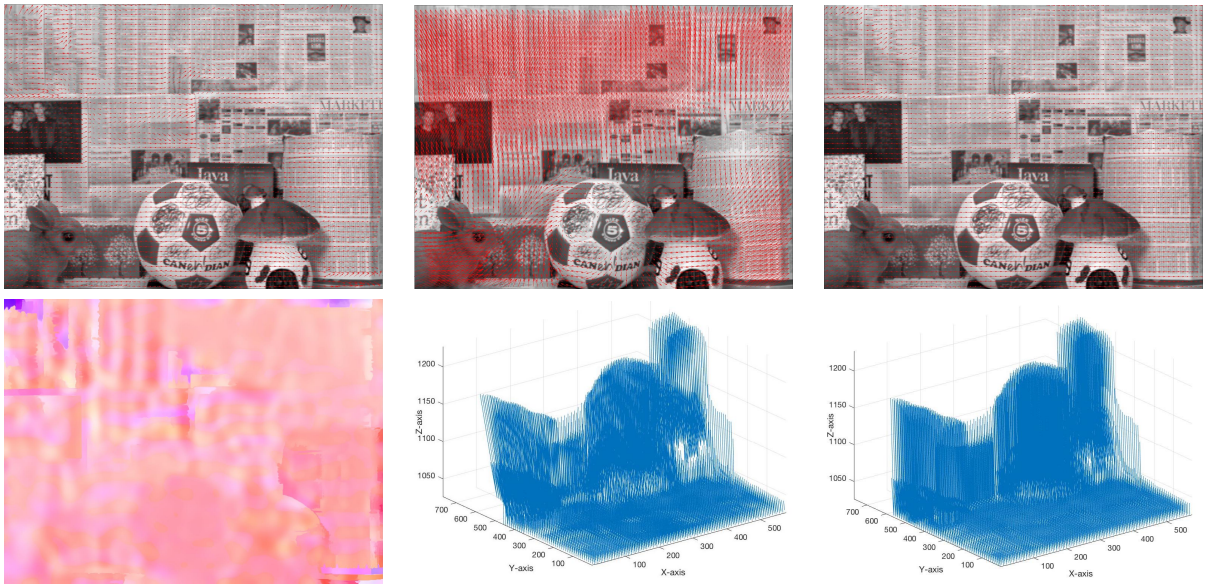
front camera had a size of 576×768. In this test, we used frames 3 and 4 in both **front_trans** and **front_div** datasets. We utilized *max* levels and $\eta = 0.95$ to build Brox pyramid.

Table 7.19 shows the 2D error measurements in terms of the flow density, AAE (Average Angular Error), EPE (End Point Error), MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and the execution time in seconds.

Figure 7.29 and Figure 7.30 show the translation motion using **CLG** and **Brox** approaches, while Figure 7.31 and Figure 7.32 represent the divergence motion using **CLG** and **Brox** approaches, respectively.

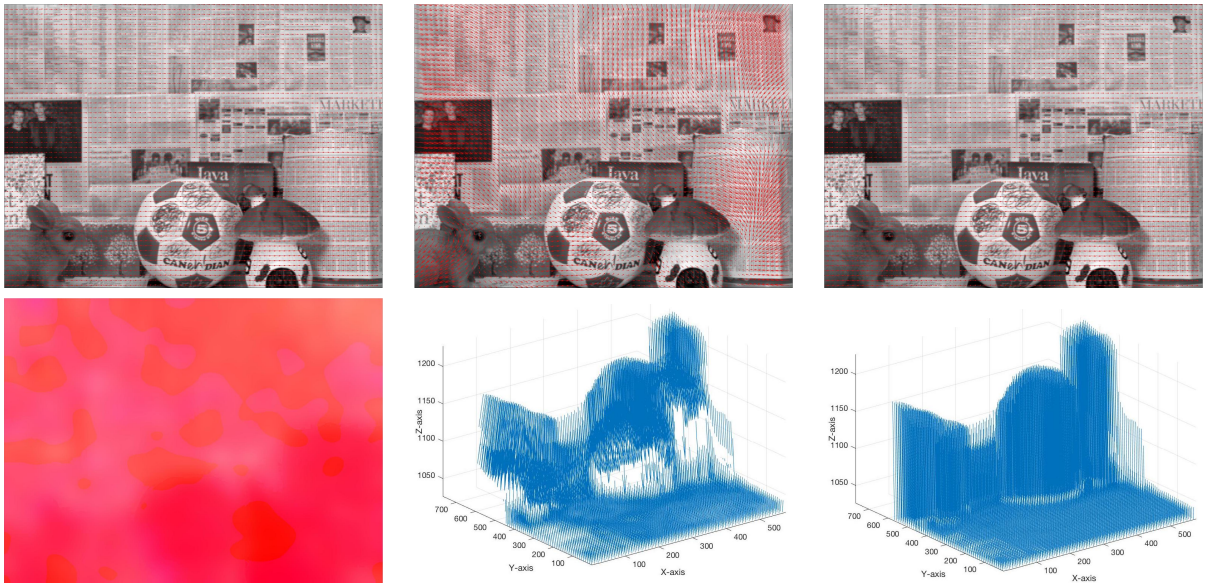
Table 7.19: Error measurements for Rear camera in iPhone X datasets

Rear_Trans						
Method	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
CLG_i	100	13.37	0.35	0.41	0.41	97.36
CLG_r	100	63.36	3.44	4.37	4.13	159.09
CLG_ir	100	11.98	0.31	0.35	0.35	332.43
Brox_i	100	10.60	0.27	0.29	0.31	257.10
Brox_r	100	40.96	1.22	1.57	1.44	330.89
Brox_ir	100	10.47	0.27	0.29	0.30	305.51
Rear_Div						
Method	**Density**	**AAE**	**EPE**	**MAE**	**RMSE**	**Time**
CLG_i	100	17.15	0.33	0.39	0.36	108.86
CLG_r	100	53.55	2.40	3.10	2.80	187.29
CLG_ir	100	16.00	0.30	0.34	0.31	211.24
Brox_i	100	15.07	0.28	0.31	0.28	238.07
Brox_r	100	34.28	0.72	0.95	0.84	368.80
Brox_ir	100	16.22	0.31	0.36	0.33	365.67



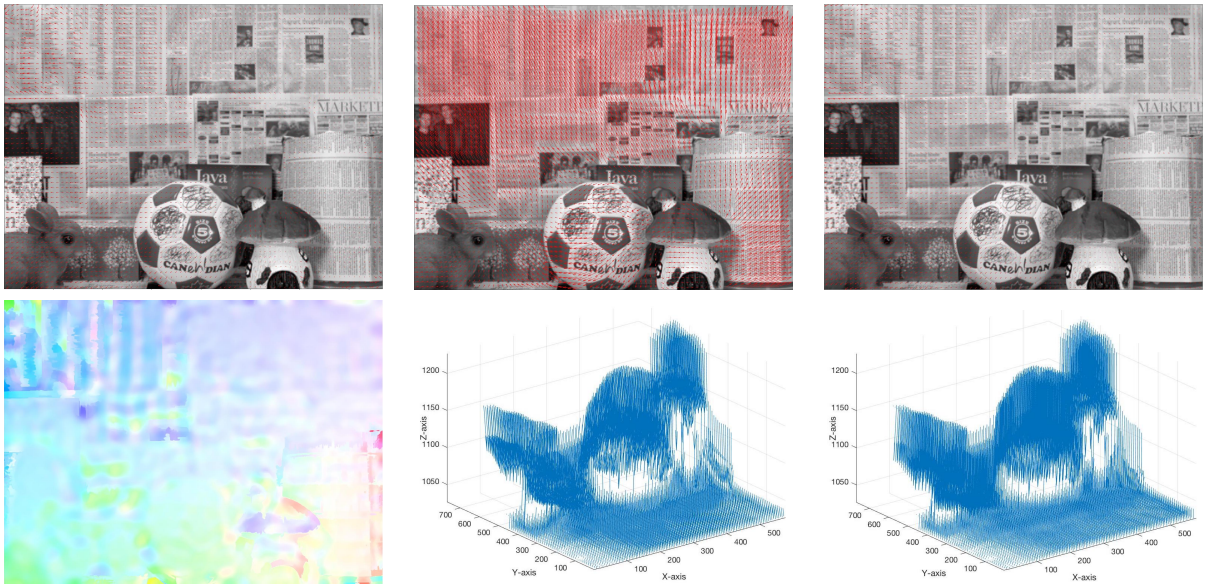
CLG_i CLG_r CLG_ir

Figure 7.29: CLG Regularization flow using Rear camera in a translation motion



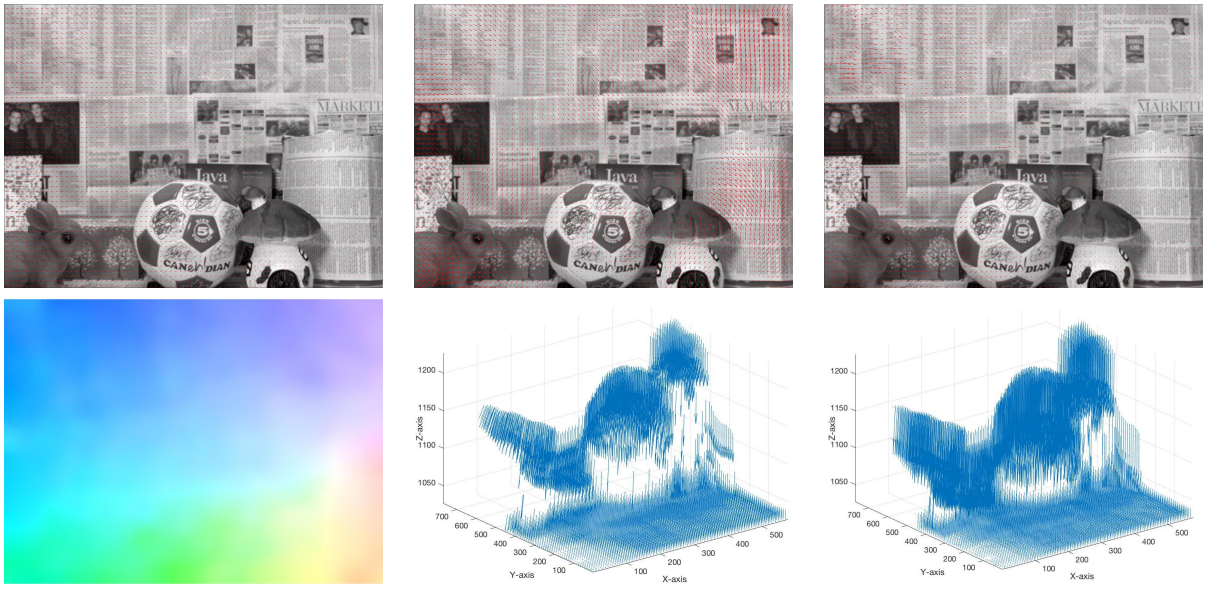
Brox_i Brox_r Brox_ir

Figure 7.30: Brox Regularization flow using Rear camera in translation motion



CLG_i CLG_r CLG_ir

Figure 7.31: CLG Regularization flow using Rear camera in divergence motion



Brox_i Brox_r Brox_ir

Figure 7.32: Brox Regularization flow using Rear camera in divergence motion

Discussion About the Cameras:
We obtained 2D optical flow and 3D range flow successfully using Kinect V2, ZED, and Truedepth camera (front camera in iPhone X) for both translation and divergence motions. However, the rear camera in iPhone X had some surprising limitations. Since we could extract the depth information from this camera, we assumed it would be fairly easy to estimate the 3D

range flow. However, intensity images from the rear camera only successfully estimated 2D optical flow. However, the depth information was not sufficient to provide adequate 3D range flow. When we restricted input to depth data only, the 3D range flow was very poor. When improved the flow by combining intensity with depth data, it nevertheless still contained had many errors. Apple has not provided enough information about their cameras for us to completely determine the cause for this, but we suspect that there is a hidden process inside the image object on the back camera in iPhone X.

7.16 Analyse Pixels Errors

In Section 4.1.13, we described another error analysis for the pixels. We computed the percentage of pixels with angle error $(10°, 25°, 50°, 75°, 95°)$. In addition, we calculated the angle error degree according to different pixel percentile (10%,25%,50%,75%,95%). We are only analysing the errors using **Brox_ir** approach with **ZED_div** as a sample.

Figure 7.33 shows the 3D pixel's error in the **Brox_ir** method when used in the **ZED_div** dataset. It also shows the AAE, EPE, MAE, RMSE, and NRMSE.

3D Flow density 100.00%
Average 3D angle error (**AAE**): 6.91
3D End Point Error (**EPE**) : 0.27
3D Mean Absolute Error (**MAE**) : 0.36
3D Root Mean Square Error (**RMSE**) : 0.28
3D Normalized Root Mean Square Error (**NRMSE**) : 0.57

Percentage of pixels with angle error less than or equal to **10** degrees: 96.71
Percentage of pixels with angle error less than or equal to **25** degrees: 100.00
Percentage of pixels with angle error less than or equal to **50** degrees: 100.00
Percentage of pixels with angle error less than or equal to **75** degrees: 100.00
Percentage of pixels with angle error less than or equal to **95** degrees: 100.00

Angle error in degrees of flow at **10** percentile: 5.63
Angle error in degrees of flow at **25** percentile: 5.91
Angle error in degrees of flow at **50** percentile: 6.52
Angle error in degrees of flow at **75** percentile: 7.60
Angle error in degrees of flow at **95** percentile: 9.16

Figure 7.33: Pixels errors for **Brox_ir** approach

7.17 General Discussion About the Results

Based on the previous experiments and results, we can conclude several points:

- The 2D optical flow algorithms can be extended to estimate 3D range flow using range data alone or combining intensity and range data together.

- Adding the pyramid framework with warping techniques helps the range flow algorithms to estimate the large motion more accurately compared to one level size image estimation.
- Our framework can estimate the 2D/3D flow up to 5*mm* displacement successfully.
- Our implemented framework can estimate adequate 3D range flow using range data only, eliminating the need for intensity data. This feature overcomes the previous estimation difficulties due to intensity illumination problems. Furthermore, we can estimate the flow even in the dark using the sensors that capture depth without intensity.
- By adding the weighted factor β to balance the derivatives using the equation mentioned in section 4.1.4, we improved the output 3D range flow noticeably.
- Applying the Weighted Median filter for the intermediate flow between the adjacent levels in the pyramid can observably improve the final 2D/3D flow.
- Using penalty statistically robust methods in our implementation can improve the AAE of the 3D output flow; however, this greatly increases the execution time. For applications that prioritize accuracy flow over execution time, these robust steps will be useful.
- Of all the approaches that we implemented, we found that **CLG** and **Brox** methods provide the best 2D/3D flow using the three different datatypes **(i, r, ir)**. However, our new approaches, including the indirect regularization approach **(Global_ind)**, also offer promising results to improve the 2D/3D flow.
- We proved that our frameworks can successfully estimate the following two motions: translation in x-direction and divergences toward the scene in the z-direction. Other motions that we did not test also have the potential to work within this framework.
- Our implementation framework estimates 2D/3D flow in about half a minute (30 seconds) based on CPU processing. This is because it can compute the flow using a single frame in a variational model.
- We tested different intensity/range sequences generated using different cameras/sensors successfully. Therefore, the range data gained using Time-Of-Flight (TOF), Structured Light (SL) and stereo-vision techniques can be used in our framework.
- This thesis provides new real datasets using Kinect V2, ZED, iPhone X (front and rear) cameras with ground truth flow. These datasets can be used to evaluate other RGB-D based optical/range flow framework.

7.18 Conclusion

In this chapter, we discussed several experiments using the implemented approaches and generated datasets. We started by demonstrating the pyramid effect and concluded with a pixel error analysis. Overall, we showed that we could produce 3D range flow using range data only, but that we could improve this flow by adding intensity data using several techniques. The final chapter presents a conclusion and discusses future work of the thesis.

Chapter 8

Conclusion and Future works

In this chapter, we draw a conclusion based on our work as explained in the previous chapters; in addition, the future works also provided at the end part.

8.1 Conclusion

Our work focused on estimating 2D/3D flow. 2D optical flow can be estimated using 2D intensity data (regular images). Calculating 3D optical flow requires 3D data, such as MRI datasets. Scene and range flow can be considered a special kind of 3D optical flow because it is defined as the 3D motion of the moving objects' surface. In this case, the algorithm requires depth information that can be gathered using several techniques, including the stereo vision method, Time-of-Flight camera, or Structured Light technique.

Using intensity data alone, we can only estimate 2D optical flow. However, combining the intensity with depth information does allow the computation of 3D range flow. Notably, the depth information in isolation also permits estimation of the 3D range flow. This feature overcomes the illumination change issues that often happen while capturing datasets. Importantly, 3D range flow can be estimated even in the dark because the sensors that measure depth typically do not require a light. Most scene flow methods fail in this case because their methods require intensity information.

This thesis studied several methods that obtain range imaging. We deliberately sought to use widely available consumer depth sensors to collect new datasets. We generated these datasets in Professor John Barron's lab at Western University using the positioner and controller; consequently, the known motions of the scene are provided. Two motions have been captured: translation in X-direction and divergence toward the objects (Z-direction). We used Kinect V2, ZED, and iPhone X (front and rear) cameras and then applied several pre-processing steps to the datasets to prepare them for the 2D/3D flow estimation algorithms.

We addressed 2D and 3D motion using several approaches. We started by applying the local methods using either (LS) and (TLS) techniques. In addition, we tried global methods that might estimate the motion directly from the derivatives or use the local approach results as

the initial stage for the regularization. We then used the combined local and global (CLG) [5] method for 2D and 3D flow estimation. Lastly, we extended Brox's approaches [6] to handle 3D range flow motions. We tested these approaches with different robust techniques to reduce the number of outliers in the estimation of optical flow and to overcome the discontinuity problem in the data. We tested the Lorentzian, Charbonnier, Generalized Charbonnier, and Geman Mcclure methods in our algorithms.

Using generated and available datasets, we tested the presented algorithms and performed a quantitative and qualitative analysis. We have been addressed the effect of using hierarchical framework with several large displacements. In addition, Local, Global, CLG, Brox approaches tested with the different robust techniques. Finally, we showed the effect of the different cameras that we utilized in estimating the 2D/3D flow. Middlebury's datasets were used to demonstrate the comparison with other scene/range flow studies.

8.2 Future Works

Future works can add colour channels (not only intensity) with depth information to estimate 3D range flow using fusion depth and intensity approaches. Although the flow might be improved, as informed by [114, 115], we expect a small improvement in the flow to be a massive process. Segmenting objects using the depth similarity and trajectory could also be an interesting task. This process would help estimate the flow for individual objects in the scene according to depth. Another possible improvement is applying a high-order polynomial expansion to the range flow approaches. Nowadays, new modern computers are released with significant GPU that help speed up the computation time required to extend these approaches. Lastly, given recent scholarly interest in deep learning concepts, re-casting the approaches described in this thesis with those deep learning techniques could also be an exciting task.

Bibliography

[1] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pp. 674–679, 1981.

[2] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *ARTIFICAL INTELLIGENCE*, vol. 17, pp. 185–203, 1981.

[3] H. Spies, B. Jhne, and J. L. Barron, "Range Flow Estimation," *Computer Vision and Image Understanding*, vol. 85, pp. 209–231, 2002.

[4] A. Bruhn, J. Weickert, and C. Schnörr, "Combining the Advantages of Local and Global Optic Flow Methods," in *Pattern Recognition* (L. Van Gool, ed.), (Berlin, Heidelberg), pp. 454–462, Springer Berlin Heidelberg, 2002.

[5] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods," *International Journal of Computer Vision*, vol. 61, pp. 211–231, Feb 2005.

[6] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High Accuracy Optical Flow Estimation Based on a Theory For Warping," in *Computer Vision-ECCV 2004*, pp. 25–36, Springer, 2004.

[7] H. Spies and J. Barron, "Range Flow : The 3D Movement of Deformable Surfaces." This article done by Dr. Barron, 2000.

[8] M. J. Black and P. Anandan, "The Robust Estimation of Multiple Motions : Parametric and Piecewise-Smooth Flow Fields," *Computer Vision and Image Understanding*, vol. 63, no. 1, pp. 75–104, 1996.

[9] D. Pagliari and L. Pinto, "Calibration of Kinect for Xbox One and Comparison Between the Two Generations of Microsoft Sensors," *Sensors (Switzerland)*, vol. 15, no. 11, pp. 27569–27589, 2015.

[10] Orbbec, "Orbbec Persee," 2018. Last visited on August,2018.

[11] F. Becker, S. Petra, and C. Schn, *Optical Flow*. Springer, 2015.

[12] D. Fortun, P. Bouthemy, and C. Kervrann, "Optical Flow Modeling and Computation: A Survey," *Computer Vision and Image Understanding*, vol. 134, pp. 1 – 21, 2015.

[13] S. Ali and M. Shah, "Human Action Recognition in Videos Using Kinematic Features and Multiple Instance Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 2, pp. 288–303, 2010.

[14] M. Jain, H. Jgou, and P. Bouthemy, "Better Exploiting Motion for Better Action Recognition," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2555–2562, June 2013.

[15] H. Wang, A. Kläser, C. Schmid, and C. L. Liu, "Dense Trajectories and Motion Boundary Descriptors for Action Recognition," *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, 2013.

[16] M. Jakubowski and G. Pastuszak, "Block-based Motion Estimation Algorithms - A Survey," *Opto-electronics Review*, vol. 21, no. 1, pp. 86–102, 2013.

[17] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank, "A Survey on Visual Content-based Video Indexing and Retrieval," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 41, no. 6, pp. 797–819, 2011.

[18] G. Piriou, P. Bouthemy, and J. . Yao, "Recognition of Dynamic Video Contents With Global Probabilistic Models of Visual Motion," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3417–3430, 2006.

[19] C. Su, H. M. Liao, H. Tyan, C. Lin, D. Chen, and K. Fan, "Motion Flow-based Video Retrieval," *IEEE Transactions on Multimedia*, vol. 9, no. 6, pp. 1193–1201, 2007.

[20] R. Gal, N. Kiryati, and N. Sochen, "Progress in the Restoration of Image Sequences Degraded by Atmospheric Turbulence," *Pattern Recognition Letters*, vol. 48, pp. 8–14, 2014.

[21] M. Werlberger, T. Pock, M. Unger, and H. Bischof, "Optical Flow Guided TV-L 1 Video Interpolation and Restoration," in *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 273–286, Springer, 2011.

[22] N. Hata, A. Nabavi, S. Warfield, W. Wells, R. Kikinis, and F. A. Jolesz, "A Volumetric Optical Flow Method for Measurement of Brain Deformation from Intraoperative Magnetic Resonance Images," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 928–935, Springer, 1999.

[23] M. Xavier, A. Lalande, P. M. Walker, F. Brunotte, and L. Legrand, "An Adapted Optical Flow Algorithm for Robust Quantification of Cardiac Wall Motion from Standard Cine-MR Examinations," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 5, pp. 859–868, 2012.

[24] T.-C. Huang, C.-K. Chang, C.-H. Liao, and Y.-J. Ho, "Quantification of Blood Flow in Internal Cerebral Artery by Optical Flow Method on Digital Subtraction Angiography in Comparison with Time-Of-Flight Magnetic Resonance Angiography," *PLOS ONE*, 2013.

[25] B. Glocker, N. Komodakis, N. Paragios, G. Tziritas, and N. Navab, "Inter and Intra-modal Deformable Registration: Continuous Deformations Meet Efficient Optimal Linear Programming," in *Biennial International Conference on Information Processing in Medical Imaging*, pp. 408–420, Springer, 2007.

[26] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach, and D. J. Hawkes, "Nonrigid Registration using Free-form Deformations: Application to Breast MR Images," *IEEE Transactions on Medical Imaging*, vol. 18, no. 8, pp. 712–721, 1999.

[27] A. Sotiras, C. Davatzikos, and N. Paragios, "Deformable Medical Image Registration: A Survey," *IEEE Transactions on Medical Imaging*, vol. 32, no. 7, pp. 1153–1190, 2013.

[28] F. Amat, E. W. Myers, and P. J. Keller, "Fast and Robust Optical Flow for Time-lapse Microscopy using Super-voxels," *Bioinformatics*, vol. 29, no. 3, pp. 373–380, 2012.

[29] D. Fortun, P. Bouthemy, P. Paul-Gilloteaux, and C. Kervrann, "Aggregation of Patch-based Estimations for Illumination-invariant Optical Flow in Live Cell Imaging," in *2013 IEEE 10th International Symposium on Biomedical Imaging*, pp. 660–663, 2013.

[30] J. Delpiano, J. Jara, J. Scheer, O. A. Ramírez, J. Ruiz-del Solar, and S. Härtel, "Performance of Optical Flow Techniques for Motion Analysis of Fluorescent Point Signals in Confocal Microscopy," *Machine Vision and Applications*, vol. 23, no. 4, pp. 675–689, 2012.

[31] K. Liu, S. S. Lienkamp, A. Shindo, J. B. Wallingford, G. Walz, and O. Ronneberger, "Optical Flow Guided Cell Segmentation and Tracking in Developing Tissue," in *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, pp. 298–301, 2014.

[32] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, June 2012.

[33] A. Giachetti, M. Campani, and V. Torre, "The Use of Optical Flow for Road Navigation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 34–48, 1998.

[34] Z. Sun, G. Bebis, and R. Miller, "On-road Vehicle Detection: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, pp. 694–711, 2006.

[35] H. Chao, Y. Gu, and M. Napolitano, "A Survey of Optical Flow Techniques for Robotics Navigation Applications," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1-4, pp. 361–372, 2014.

[36] A. Crtual and F. Chaumette, "Visual Servoing Based on Image Motion," *The International Journal of Robotics Research*, vol. 20, no. 11, pp. 857–877, 2001.

[37] W. Enkelmann, "Obstacle Detection by Evaluation of Optical Flow Fields from Image Sequences," *Image and Vision Computing*, vol. 9, no. 3, pp. 134–138, 1991.

[38] M. J. Black and Y. Yacoob, "Recognizing Facial Expressions in Image Sequences using Local Parameterized Models of Image Motion," *International Journal of Computer Vision*, vol. 25, no. 1, pp. 23–48, 1997.

[39] R. Cutler and M. Turk, "View-based Interpretation of Real-time Optical Flow for Gesture Recognition," in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 416–421, 1998.

[40] A. Basset, P. Bouthemy, and C. Kervrann, "Recovery of Motion Patterns and Dominant Paths in Videos of Crowded Scenes," in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 184–188, 2014.

[41] N. Kiryati, T. R. Raviv, Y. Ivanchenko, and S. Rochel, "Real-time Abnormal Motion Detection in Surveillance Video," in *2008 19th International Conference on Pattern Recognition*, pp. 1–4, 2008.

[42] P. Zanuttigh, G. Marin, C. Dal Mutto, F. Dominio, L. Minto, and G. M. Cortelazzo, *Time-of-Flight and Structured Light Depth Cameras: Technology and Applications*. Springer, 2016.

[43] R. Crabb, C. Tracey, A. Puranik, and J. Davis, "Real-time Foreground Segmentation via Range and Color Imaging," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–5, IEEE, 2008.

[44] O. Wang, J. Finger, Q. Yang, J. Davis, and R. Yang, "Automatic Natural Video Matting with Depth," in *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, pp. 469–472, 2007.

[45] T. Lu and S. Li, "Image Matting with Color and Depth Information," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 3787–3790, 2012.

[46] L. Wang, C. Zhang, R. Yang, and C. Zhang, "Tofcut: Towards Robust Real-time Foreground Extraction Using a Time-Of-Flight Camera," in *Proc. of 3DPVT*, pp. 1–8, 2010.

[47] C. Stauffer and W. E. L. Grimson, "Learning Patterns of Activity Using Real-time Tracking," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 747–757, 2000.

[48] B. Langmann, S. E. Ghobadi, K. Hartmann, and O. Loffeld, "Mulie-model Backgound Subtraction Using Gaussian Mixture Models," 2010.

[49] M. Camplani and L. Salgado, "Background Foreground Segmentation with RGB-D Kinect data: An Efficient Combination of Classifiers," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 122–136, 2014.

[50] A. Störmer, M. Hofmann, and G. Rigoll, "Depth Gradient Based Segmentation of Overlapping Foreground Objects in Range Images," in *2010 13th international conference on information fusion*, pp. 1–4, 2010.

[51] C. Dal Mutto, P. Zanuttigh, and G. M. Cortelazzo, "Fusion of Geometry and Color Information for Scene Segmentation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 5, pp. 505–521, 2012.

[52] B. Dellen, G. Alenya, S. Foix, and C. Torras, "Segmenting Color Images Into Surface Patches by Exploiting Sparse Depth Data," in *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, pp. 591–598, 2011.

[53] F. Calderero and F. Marques, "Hierarchical Fusion of Color and Depth Information at Partition Level by Cooperative Region Merging," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 973–976, IEEE, 2009.

[54] J. Leens, S. Piérard, O. Barnich, M. Van Droogenbroeck, and J.-M. Wagner, "Combining Color, Depth, and Motion for Video Segmentation," in *International Conference on Computer Vision Systems*, pp. 104–113, Springer, 2009.

[55] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *European Conference on Computer Vision*, pp. 746–760, Springer, 2012.

[56] A. Hermans, G. Floros, and B. Leibe, "Dense 3d Semantic Mapping of Indoor Scenes from RGBD Images," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2631–2638, 2014.

[57] A. C. Müller and S. Behnke, "Learning Depth-sensitive Conditional Random Fields for Semantic Segmentation of RGB-D Images," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6232–6237, 2014.

[58] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient Regression of General-activity Human Poses from Depth Images," in *2011 International Conference on Computer Vision*, pp. 415–422, IEEE, 2011.

[59] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time Human Pose Recognition in Parts from Single Depth Images," in *CVPR 2011*, pp. 1297–1304, 2011.

[60] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys, "Accurate 3D Pose Estimation from a Single Depth Image," in *2011 International Conference on Computer Vision*, pp. 731–738, 2011.

[61] J. Suarez and R. R. Murphy, "Hand Gesture Recognition with Depth Images: A Review," in *2012 IEEE RO-MAN: the 21st IEEE international symposium on robot and human interactive communication*, pp. 411–417, 2012.

[62] T. Kapuscinski, M. Oszust, M. Wysocki, and D. Warchol, "Recognition of Hand Gestures Observed by Depth Cameras," *International Journal of Advanced Robotic Systems*, vol. 12, no. 4, p. 36, 2015.

[63] P. Suryanarayan, A. Subramanian, and D. Mandalapu, "Dynamic Hand Pose Recognition Using Depth Data," in *2010 20th International Conference on Pattern Recognition*, pp. 3105–3108, 2010.

[64] L. Nanni, A. Lumini, F. Dominio, M. Donadeo, and P. Zanuttigh, "Combination of Depth and Texture Descriptors for Gesture Recognition," 2014.

[65] J. Lang and D. K. Pai, "Estimation of Elastic Constants from 3D Range Flow," *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM*, vol. 2001-January, pp. 331–338, 2001.

[66] H. Spies, B. Jähne, and J. L. Barron, "Surface Expansion from Range Data Sequences," in *Pattern Recognition* (B. Radig and S. Florczyk, eds.), (Berlin, Heidelberg), pp. 163–169, Springer Berlin Heidelberg, 2001.

[67] J. Barron, A. Liptay, and H. Spies, "Optical and Range Flow to Measure 3D Plant Growth and Motion," *Image and Vision Computing'2000 (ICVNZ2000)*, 2000.

[68] A. Liptay and J. L. Barron, "Real-Time , Non-Contact Plant Growth Monitoring at Microscopic Levels using 3D Laser Scanner .," *October*, pp. 3–7, July 2014.

[69] H. Spies and J. Barron, "Estimating Expansion Rates from Range Data Sequences," 2002.

[70] X. Tang, J. L. Barron, R. E. Mercer, and P. Joe, "Tracking Weather Storms Using 3D Doppler Radial Velocity Information," in *Scandinavian Conference on Image Analysis*, pp. 1038–1043, Springer, 2003.

[71] S. Ghuffar, B. Szkely, A. Roncat, and N. Pfeifer, "Landslide Displacement Monitoring using 3D Range Flow on Airborne and Terrestrial LiDAR Data," *Remote Sensing*, vol. 5, no. 6, pp. 2720–2745, 2013.

[72] A. Roncat, S. Ghuffar, B. Székely, P. Dorninger, S. Rasztovits, M. Mittelberger, Z. Koma, D. Krawczyk, and N. Pfeifer, "A Natural Laboratory Terrestrial Laser Scanning and auxiliary Measurements for studying an active Landslide," September 2013.

[73] M. Jaimez and J. Gonzalez-Jimenez, "Fast Visual Odometry for 3-D Range Sensors," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 809–822, 2015.

[74] M. Jaimez, J. G. Monroy, and J. Gonzalez-Jimenez, "Planar Odometry from a Radial Laser Scanner. A Range Flow-based Approach," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 4479–4485, 2016.

[75] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers, "Fast Odometry and Scene Flow from RGB-D Cameras Based on Geometric Clustering," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3992–3999, May 2017.

[76] M. Jaimez, J. Monroy, M. Lopez-Antequera, and J. Gonzalez-Jimenez, "Robust Planar Odometry Based on Symmetric Range Flow and Multi-Scan Alignment," *IEEE Transactions on Robotics*, vol. 34, pp. 1–13, 2018.

[77] K. Al Ismaeil, D. Aouada, T. Solignac, B. Mirbach, and B. Ottersten, "Real-Time Enhancement of Dynamic Depth Videos with Non-Rigid Deformations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 10, pp. 2045–2059, 2016.

[78] J. Nebel and A. Sibiryakov, "Range Flow from Stereo-temporal Matching: Application to Skinning," *IASTED Int. Conf. on Visualization, Imaging, and Image Processing*, vol. 2, no. 2.2, p. 2, 2002.

[79] J. L. Barron, "Experience With 3D Optical Flow on Gated MRI Cardiac Datasets," in *First Canadian Conference on Computer and Robot Vision, 2004. Proceedings.*, pp. 370–377, IEEE, 2004.

[80] I. Ravyse and H. Sahli, "Facial Analysis and Synthesis Scheme," in *Advanced Concepts for Intelligent Vision Systems* (J. Blanc-Talon, W. Philips, D. Popescu, and P. Scheunders, eds.), (Berlin, Heidelberg), pp. 810–820, Springer Berlin Heidelberg, 2006.

[81] I. Ravyse and H. Salhli, "A Biomechanical Model for Image-based Estimation of 3D Face Deformations," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1089–1092, 2008.

[82] H. Gharavi and S. Gao, "3-D Motion Estimation Using Range Data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 133–143, 2007.

[83] M. Haindl, P. Žid, and R. Holub, "Range Video Segmentation," *10th International Conference on Information Sciences, Signal Processing and their Applications, ISSPA 2010*, pp. 369–372, 2010.

[84] Z. Fang, S. Yang, S. Jain, G. Dubey, S. Roth, S. Maeta, S. Nuske, Y. Zhang, and S. Scherer, "Robust Autonomous Flight in Constrained and Visually Degraded Shipboard Environments," *Journal of Field Robotics*, vol. 34, no. 1, pp. 25–52, 2017.

[85] M. Yagcoglu, K. Yelen, and A. Temizel, "Integrated Object Tracking Framework for Range Gated Camera Systems," *Long-Range Imaging*, vol. 9846, no. May 2016, p. 984607, 2016.

[86] C. D. Monaco and S. N. Brennan, "Ego-Motion Estimate Corruption Due to Violations of the Range Flow Constraint," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3964–3969, 2018.

[87] M. Menze and A. Geiger, "Object Scene Flow for Autonomous Vehicles," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3061–3070, June 2015.

[88] H. Spies, H. Haußecker, B. Jähne, and J. L. Barron, "Differential Range Flow Estimation," in *Mustererkennung 1999* (W. Förstner, J. M. Buhmann, A. Faber, and P. Faber, eds.), (Berlin, Heidelberg), pp. 309–316, Springer Berlin Heidelberg, 1999.

[89] H. Spies, J. Bernd, and J. L. Barron, "Regularised Range Flow," *ECCV*, pp. 785–799, 2000.

[90] D. Sun, S. Roth, and M. J. Black, "A Quantitative Analysis of Current Practices in Optical Flow Estimation and The Principles Behind Them," *International Journal of Computer Vision*, vol. 106, no. 2, pp. 115–137, 2014.

[91] D. Scharstein and R. Szeliski, "High-accuracy Stereo Depth Maps Using Structured Light," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 1, 2003.

[92] J. L. Barron and N. A. Thacker, "Tutorial: Computing 2D and 3D optical flow," *Imaging Science and Biomedical Engineering Division, Medical School, University of Manchester*, 2005.

[93] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud, "Two Deterministic Half-quadratic Regularization Algorithms for Computed Imaging," *Proceedings - International Conference on Image Processing, ICIP*, vol. 2, no. 3, pp. 168–172, 1994.

[94] W. Chen and J. L. Barron, "High Accuracy Optical Flow Method Based on a Theory for Warping: 3D Extension," in *International Conference Image Analysis and Recognition*, pp. 250–262, Springer, 2010.

[95] B. Horn and J. Harris, "Rigid Body Motion from Range Image Sequences," *CVGIP: Image Understanding*, vol. 53, no. 1, pp. 1–13, 1991.

[96] H. Spies and J. Barron, "Evaluating the Range Flow Motion Constraint," *Object recognition supported by user interaction for service robots*, vol. 3, no. c, pp. 517–520, 2002.

[97] H. Spies, B. Jahne, B. J. De, and J. L. Barron, "Dense Range Flow from Depth and Intensity Data," *Pattern Recognition, 2000. Proceedings. 15th International Conference*, vol. 1, pp. 131–134, 2000.

[98] J. L. Barron and H. Spies, "Quantitative Regularized Range Flow," *In Vision Interface*, pp. 203–210, 2000.

[99] J. Barron and H. Spies, "The Fusion of Image and Range Flow," *Multi-Image Analysis*, pp. 171–189, 2001.

[100] M. Yamamoto, P. Boulanger, and M. Rioux, "Direct Estimation of Range Flow on Deformable Shape from a Video Rate Range Camera," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 15, no. 1, pp. 82–89, 1993.

[101] T. Darrell, G. Gordon, J. Woodfill, P. M. Road, and P. Alto, "3D Pose Tracking with Linear Depth and Brightness Constraints," *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1, pp. 206 – 213, 1999.

[102] A. Imiya and D. Yamada, "Voting Method for Stable Range Optical Flow Computation," *Lecture Notes in Computer Science*, pp. 332–341, 2006.

[103] S. Matzka, Y. R. Petillot, and A. M. Wallace, "Fast Motion Estimation on Range Image Sequences Acquired with a 3-D Camera," in *BMVC*, pp. 1–10, 2007.

[104] M. Schmidt, M. Jehle, and B. Jahne, "Range Flow Estimation Based on Photonic Mixing Device Data," *International Journal of Intelligent Systems Technologies and Applications*, vol. 5, no. 3-4, pp. 380–392, 2008.

[105] T. Schuchert, T. Aach, and H. Scharr, "Range Flow for Varying Illumination," *Eccv*, pp. 509–522, 2008.

[106] T. Schuchert, T. Aach, and H. Scharr, "Range Flow in Varying Illumination: Algorithms and Comparisons," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1646–1658, 2010.

[107] T. Schuchert and H. Scharr, "Estimation of 3D Object Structure, Motion and Rotation Based on 4D Affine Optical Flow Using a Multi-Camera Array," in *European Conference on Computer Vision*, pp. 596–609, Springer, 2010.

[108] T. Schuchert and H. Scharr, "Simultaneous Estimation of Surface Motion, Depth and Slopes Under Changing Illumination," in *Joint Pattern Recognition Symposium*, pp. 184–193, Springer, 2007.

[109] T. Schuchert and H. Scharr, "An Affine Optical Flow Model for Dynamic Surface Reconstruction," in *Statistical and Geometrical Approaches to Visual Motion Analysis*, pp. 70–90, Springer, 2009.

[110] T. Yu and J. Lang, "Window-based Range Flow With an Isometry Constraint," *CRV 2010 - 7th Canadian Conference on Computer and Robot Vision*, pp. 331–338, 2010.

[111] J. M. Gottfried, J. Fehr, and C. S. Garbe, "Computing Range Flow from Multi-modal Kinect Data," *ISVC 2011*, vol. 6938 LNCS, no. PART 1, pp. 758–767, 2010.

[112] S. L. Francis, S. G. Anavatti, and M. Garratt, "Range Based Velocity Estimation Using Scene Flow," *Proceedings of the 2012 International Conference on Artificial Intelligence, ICAI 2012*, vol. 2, pp. 1–4, 2012.

[113] S. Ghuffar, N. Brosch, N. Pfeifer, and M. Gelautz, "Motion Segmentation in Videos from Time of Flight Cameras," *2012 19th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 328–332, 2012.

[114] T. C. Lukins and R. B. Fisher, "Colour Constrained 4D Flow.," in *BMVC*, 2005.

[115] T. Birdal, D. Mateus, and S. Ilic, "Towards A Complete Framework For Deformable Surface Recovery Using RGBD Cameras," in *International Robots and Systems (IRoS), Workshop on Color-Depth Fusion in Robotics*, (Vila Moura, Portugal), Oct. 2012.

[116] G. A. Jones, "Accurate and Computationally-inexpensive Recovery of Ego-Motion using Optical Flow and Range Flow with Extended Temporal Support," *Bmvc2013*, pp. 1–10, 2013.

[117] G. Jones, "Combining Optical Flow and Range Flow to Recover RGBD Sensor Egomotion," in *RGB-D workshop at Robotics Science and Systems (RSS) Conference, Berlin, Germany*, vol. 80, pp. 1–6, Citeseer, 2013.

[118] G. A. Jones and G. Hunter, "Spatio-temporal Support for Range Flow Based Ego-Motion Estimators," in *Computer Analysis of Images and Patterns* (R. Wilson, E. Hancock, A. Bors, and W. Smith, eds.), (Berlin, Heidelberg), pp. 531–538, Springer Berlin Heidelberg, 2013.

[119] S. Ghuffar, C. Ressl, and N. Pfeifer, "Relative Orientation of Videos from Range Imaging Cameras," in *Videometrics, Range Imaging, and Applications XII; and Automated Visual Inspection*, vol. 8791, pp. 257–263, SPIE, 2013.

[120] S. Ghuffar, N. Brosch, N. Pfeifer, and M. Gelautz, "Motion Estimation and Segmentation in Depth and Intensity Videos," *Integrated Computer-Aided Engineering*, vol. 21, no. 3, pp. 203–218, 2014.

[121] B. Okorn and J. Harguess, "Range Image Flow using High-Order Polynomial Expansion," September 2013.

[122] B. Okorn and J. Harguess, "Ego-motion Estimation on Range Images Using High-order Polynomial Expansion," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 299–306, 2014.

[123] E. Herbst, X. Ren, and D. Fox, "RGB-D Flow: Dense 3-D Motion Estimation Using Color and Depth," *IEEE International Conference on Robotics and Automation*, pp. 2276–2282, 2013.

[124] J. Quiroga, F. Devernay, and J. Crowley, "Scene Flow by Tracking in Intensity and Depth Data," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 50–57, June 2012.

[125] J. Quiroga, F. Devernay, and J. Growley, "Local / Global Scene Flow Estimation," *Icip 2013*, pp. 1–5, 2013.

[126] J. Quiroga, *Scene Flow Estimation from RGBD Images*. PhD thesis, Grenoble University, 11 2014.

[127] J. Quiroga, F. Devernay, and J. Crowley, "Local Scene Flow by Tracking in Intensity and Depth," *J. Vis. Comun. Image Represent.*, vol. 25, pp. 98–107, Jan. 2014.

[128] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, and D. Cremers, "A primal-dual Framework for Real-time Dense RGB-D Scene Flow," *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 98–104, 2015.

[129] M. Jaimez, M. Souiai, J. Stuckler, J. Gonzalez-Jimenez, and D. Cremers, "Motion Cooperation: Smooth Piece-wise Rigid Scene Flow from RGB-D Images," *3D Vision (3DV), 2015 International Conference on*, pp. 64–72, 2015.

[130] X. Xiang, M. Zhai, R. Zhang, W. Xu, and A. El Saddik, "Scene Flow Estimation Based on 3D Local Rigidity Assumption and Depth Map Driven Anisotropic Smoothness," *IEEE Access*, vol. 6, pp. 30012–30023, 2018.

[131] T. Kanade, S. Baker, R. Collins, P. Rander, and S. Vedula, "Three-dimensional Scene Flow," in *IEEE International Conference on Computer Vision*, vol. 2, (Los Alamitos, CA, USA), p. 722, IEEE Computer Society, sep 1999.

[132] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, "Three-dimensional Scene Flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 475–480, 2005.

[133] Y. Zhang and C. Kambhamettu, "Integrated 3D Scene Flow and Structure Recovery from Multiview Image Sequences," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000*, vol. 2, pp. 674–681, 2000.

[134] Y. Zhang and C. Kambhamettu, "On 3D scene Flow and Structure Estimation," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 2, pp. II–778–II–785, 2001.

[135] Y. Zhang and C. Kambhamettu, "On 3-D Scene Flow and Structure Recovery from Multiview Image Sequences," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 33, no. 4, pp. 592–600, 2003.

[136] J. Pons and R. Keriven, "Variational Stereovision and 3D Scene Flow Estimation with Statistical Similarity Measures," *Computer Vision, 2003*, pp. 3–8, 2003.

[137] J. P. Pons, R. Keriven, and O. Faugeras, "Multi-view Stereo Reconstruction and Scene Flow Estimation With a Global Image-based Matching Score," *International Journal of Computer Vision*, vol. 72, no. 2, pp. 179–193, 2007.

[138] M. Isard and J. MacCormick, "Dense Motion and Disparity Estimation via Loopy Belief Propagation," *ACCV 2006*, pp. 32–41, 2006.

[139] F. Devernay, D. Mateus, and M. Guilbert, "Multi-camera Scene Flow by Tracking 3-D Points and Surfels," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 2203–2212, 2006.

[140] F. Huguet and F. Devernay, "A Variational Method for Scene Flow Estimation from Stereo Sequences," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1–7, 2007.

[141] R. Li and S. Sclaroff, "Multi-scale 3D Scene Flow from Binocular Stereo Sequences," *Computer Vision and Image Understanding*, vol. 110, no. 1, pp. 75–90, 2008.

[142] A. Wedel, T. Brox, T. Vaudrey, C. Rabe, U. Franke, and D. Cremers, "Stereoscopic Scene Flow Computation for 3D Motion Understanding," *International Journal of Computer Vision*, vol. 95, no. 1, pp. 29–51, 2010.

[143] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers, "Efficient Dense Scene Flow from Sparse or Dense Stereo Data," *Lecture Notes in Computer Science*, vol. 5302 LNCS, pp. 739–751, 2008.

[144] A. Wedel and D. Cremers, *Stereo Scene Flow for 3D Motion Analysis*. Springer Publishing Company, Incorporated, 1st ed., 2011.

[145] A. Wedel, A. Meißner, C. Rabe, U. Franke, and D. Cremers, "Detection and Segmentation of Independently Moving Objects from Dense Scene Flow," *EMMCVPR*, vol. 5681 LNCS, pp. 14–27, 2009.

[146] C. Rabe, T. Müller, A. Wedel, and U. Franke, "Dense, Robust, and Accurate Motion Field Estimation from Stereo Image Sequences in Real-Time," in *Computer Vision - ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), pp. 582–595, Springer Berlin Heidelberg, 2010.

[147] L. Valgaerts, A. Bruhn, H. Zimmer, J. Weickert, C. Stoll, and C. Theobalt, "Joint Estimation of Motion, Structure and Geometry from Stereo Sequences," in *Computer Vision – ECCV 2010*, pp. 568–581, Springer Berlin Heidelberg, 2010.

[148] J. Čech, J. Sanchez-Riera, and R. Horaud, "Scene Flow Estimation by Growing Correspondence Seeds," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3129–3136, 2011.

[149] C. Vogel, K. Schindler, and S. Roth, "3D Scene Flow Estimation With a Rigid Motion Prior," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1291–1298, 2011.

[150] S. Hadfield and R. Bowden, "Kinecting the Dots: Particle Based Scene Flow from Depth Sensors," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2290–2295, 2011.

[151] A. Letouzey, B. Petit, and E. Boyer, "Scene Flow from Depth and Color Images," in *BMVC 2011 - British Machine Vision Conference*, Proceedings of the British Machine Vision Conference, pp. 46:1–11, BMVA Press, 2011.

[152] M. Sizintsev and R. P. Wildes, "Spatiotemporal Stereo and Scene flow Via Stequel Matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 6, pp. 1206–1219, 2012.

[153] T. Basha, Y. Moses, and N. Kiryati, "Multi-view Scene Flow Estimation: A view Centered Variational Approach," *International Journal of Computer Vision*, vol. 101, no. 1, pp. 6–21, 2013.

[154] C. Vogel, K. Schindler, and S. Roth, "Piecewise Rigid Scene Flow," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1377–1384, 2013.

[155] C. Vogel, K. Schindler, and S. Roth, "3D Scene Flow Estimation with a Piecewise Rigid Scene Model," *International Journal of Computer Vision*, vol. 115, pp. 1–28, Oct 2015.

[156] S. Hadfield and R. Bowden, "Scene Particles: Unregularized Particle-Based Scene Flow Estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 564–576, March 2014.

[157] D. Sun, E. B. Sudderth, and H. Pfister, "Layered RGBD Scene Flow Estimation," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 548–556, 2015.

[158] A. Zanfir and C. Sminchisescu, "Large Displacement 3D Scene Flow With Occlusion Reasoning," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[159] C. Richardt, H. Kim, L. Valgaerts, and C. Theobalt, "Dense Wide-Baseline Scene Flow from Two Handheld Video Cameras," in *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 276–285, Oct 2016.

[160] Z. Lv, C. Beall, P. F. Alcantarilla, F. Li, Z. Kira, and F. Dellaert, "A Continuous Optimization Approach for Efficient and Accurate Scene Flow," *CoRR*, vol. abs/1607.07983, 2016.

[161] T. Taniai, S. N. Sinha, and Y. Sato, "Fast Multi-frame Stereo Scene Flow with Motion Segmentation," *CoRR*, vol. abs/1707.01307, 2017.

[162] D. Xiao, Q. Yang, B. Yang, and W. Wei, "Monocular Scene Flow Estimation Via Variational Method," *Multimedia Tools and Applications*, vol. 76, pp. 10575–10597, Apr 2017.

[163] Z. Ren, D. Sun, J. Kautz, and E. B. Sudderth, "Cascaded Scene Flow Prediction using Semantic Segmentation," *CoRR*, vol. abs/1707.08313, 2017.

[164] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning Scene Flow in 3D Point Clouds," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[165] M. Menze, C. Heipke, and A. Geiger, "Object Scene Flow," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 140, pp. 60 – 76, 2018. Geospatial Computer Vision.

[166] X. Xiang, D. Xiao, M. Zhai, and R. Zhang, "A method of scene flow estimation with bilateral filter and adaptive TV(Total Variation) penalty function," in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pp. 883–887, 2017.

[167] R. Schuster, O. Wasenmüller, G. Kuschk, C. Bailer, and D. Stricker, "SceneFlowFields: Dense Interpolation of Sparse Scene Flow Correspondences," *CoRR*, vol. abs/1710.10096, 2017.

[168] R. Schuster, C. Bailer, O. Wasenmüller, and D. Stricker, "Combining Stereo Disparity and Optical Flow for Basic Scene Flow," *CoRR*, vol. abs/1801.04720, 2018.

[169] C. Bailer, B. Taetz, and D. Stricker, "Flow Fields: Dense Correspondence Fields for Highly Accurate Large Displacement Optical Flow Estimation," *CoRR*, vol. abs/1508.05151, 2015.

[170] H. Hirschmuller, "Stereo Processing by Semiglobal Matching and Mutual Information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 328–341, Feb 2008.

[171] R. Schuster, O. Wasenmüller, C. Unger, G. Kuschk, and D. Stricker, "SceneFlowFields++: Multi-frame Matching, Visibility Prediction, and Robust Interpolation for Scene Flow Estimation," *CoRR*, vol. abs/1902.10099, 2019.

[172] H. Haußecker, C. Garbe, H. Spies, and B. Jähne, "A Total Least Squares Framework for Low-Level Analysis of Dynamic Scenes and Processes," in *Mustererkennung 1999*, pp. 240–249, Springer Berlin Heidelberg, 1999.

[173] B. Jähne, H. Haussecker, H. Scharr, H. Spies, D. Schmundt, and U. Schurr, "Study of Dynamical Processes with Tensor-based Spatiotemporal Image Processing Techniques," in *European Conference on Computer Vision*, pp. 322–336, Springer, 1998.

[174] E. W. Weisstein, "Successive Over Relaxation.From MathWorld—A Wolfram Web Resource." Last visited on 20/7/2019.

[175] E. P. Simoncelli, "Design of Multi-dimensional Derivative Filters," in *Proceedings of 1st International Conference on Image Processing*, vol. 1, pp. 790–794 vol.1, Nov 1994.

[176] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers, "An Improved Algorithm for TV-L 1 Optical Flow," in *Statistical and Geometrical Approaches to Visual Motion Analysis*, 2008.

[177] D. Sun, "Secrets of Optical Flow Estimation and Their Principles," *Cvpr*, pp. 2432–2439, 2010.

[178] Y. Li and S. Osher, "A New Median Formula with Applications to PDE Based Denoising," *Commun. Math. Sci.*, vol. 7, pp. 741–753, 09 2009.

[179] M. J. Black and P. Anandan, "A Framework for the Robust Estimation of Optical Flow," in *1993 (4th) International Conference on Computer Vision*, pp. 231–236, May 1993.

[180] Z. Yan and X. Xiang, "Scene Flow Estimation: A Survey," *ArXiv*, vol. abs/1612.02590, 2016.

[181] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of Optical Flow Techniques," *International Journal of Computer Vision*, vol. 1, no. 12, pp. 43–77, 1994.

[182] L. Yang, L. Zhang, H. Dong, A. Alelaiwi, and A. El Saddik, "Evaluating and Improving the Depth Accuracy of Kinect For Windows v2," *IEEE Sensors Journal*, vol. 15, no. 8, pp. 4275–4285, 2015.

[183] M. Otesteanu and V. Gui, "3D Image Sensors, an Overview," *WSEAS Transactions on Electronics*, vol. 5, no. 3, pp. 53–56, 2008.

[184] G. Sansoni, M. Trebeschi, and F. Docchio, "State-of-the-art and Applications of 3D Imaging Sensors in Industry, Cultural Heritage, Medicine, and Criminal Investigation," *Sensors*, vol. 9, no. 1, pp. 568–601, 2009.

[185] H. Sarbolandi, D. Lefloch, and A. Kolb, "Kinect Range Sensing: Structured-light versus Time-of-Flight Kinect," *Computer vision and image understanding*, vol. 139, pp. 1–20, 2015.

[186] B. Billiot, F. Cointault, L. Journaux, J.-C. Simon, and P. Gouton, "3D Image Acquisition System Based on Shape from Focus Technique," *Sensors*, vol. 13, no. 4, pp. 5040–5053, 2013.

[187] T. Vaudrey, C. Rabe, R. Klette, and J. Milburn, "Differences Between Stereo and Motion Behaviour on Synthetic and Real-world Stereo Sequences," in *2008 23rd International Conference Image and Vision Computing New Zealand*, pp. 1–6, Nov 2008.

[188] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A Naturalistic Open Source Movie for Optical Flow Evaluation," in *Computer Vision-ECCV*, (Berlin, Heidelberg), pp. 611–625, 2012.

[189] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation," *CoRR*, vol. abs/1512.02134, 2015.

[190] J. Ballester and C. Pheatt, "Using the Xbox Kinect Sensor for Positional Data Acquisition," *American journal of Physics*, vol. 81, no. 1, pp. 71–77, 2013.

[191] Z. Zhang, "Microsoft Kinect Sensor and Its Effect," *IEEE multimedia*, vol. 19, no. 2, pp. 4–10, 2012.

[192] N. Kitsunezaki, E. Adachi, T. Masuda, and J.-i. Mizusawa, "Kinect Applications for the Physical Rehabilitation," in *2013 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, pp. 294–299, 2013.

[193] C.-Y. Chang, B. Lange, M. Zhang, S. Koenig, P. Requejo, N. Somboon, A. Sawchuk, and A. Rizzo, "Towards Pervasive Physical Rehabilitation Using Microsoft Kinect," in *2012 6th international conference on pervasive computing technologies for healthcare*, pp. 159–162, 2012.

[194] A. Nghiem, E. Auvinet, and J. Meunier, "Head Detection Using Kinect Camera and Its Application to Fall Detection," in *2012 11th International Conference on Information Science, Signal Processing and their Applications*, pp. 164–169, 2012.

[195] S. Ganesan and L. Anthony, "Using the Kinect to Encourage Older Adults to Exercise: A Prototype," in *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, (New York, NY, USA), pp. 2297–2302, ACM, 2012.

[196] A. Yargıç and M. Doğan, "A Lip Reading Application on MS Kinect Camera," in *2013 IEEE INISTA*, pp. 1–5, 2013.

[197] A. Agarwal and M. K. Thakur, "Sign Language Recognition Using Microsoft Kinect," in *2013 Sixth International Conference on Contemporary Computing (IC3)*, pp. 181–185, 2013.

[198] F. Marinello, A. Pezzuolo, F. Gasparini, J. Arvidsson, and L. Sartori, "Application of the Kinect Sensor for Dynamic Soil Surface Characterization," *Precision agriculture*, vol. 16, no. 6, pp. 601–612, 2015.

[199] I. P. T. Weerasinghe, J. Y. Ruwanpura, J. E. Boyd, and A. Habib, "Application of Microsoft Kinect Sensor for Tracking Construction Workers," in *Construction Research Congress 2012: Construction Challenges in a Flat World*, pp. 858–867, 2012.

[200] K. Qian, H. Yang, and J. Niu, "Developing a Gesture Based Remote Human-robot Interaction System Using Kinect," *International Journal of Smart Home*, vol. 7, no. 4, pp. 203–208, 2013.

[201] K. K. Biswas and S. K. Basu, "Gesture Recognition Using Microsoft kinect®," in *The 5th international conference on automation, robotics and applications*, pp. 100–103, 2011.

[202] D. S. Alexiadis, P. Kelly, P. Daras, N. E. O'Connor, T. Boubekeur, and M. B. Moussa, "Evaluating a Dancer's Performance Using Kinect-based Skeleton Tracking," in *Proceedings of the 19th ACM International Conference on Multimedia*, pp. 659–662, ACM, 2011.

[203] A. Abramov, K. Pauwels, J. Papon, F. Wörgötter, and B. Dellen, "Depth-supported Real-time Video Segmentation With the Kinect," in *2012 IEEE workshop on the applications of computer vision (WACV)*, pp. 457–464, 2012.

[204] A. Anwer, "Saving Kinects Depth image in 16-bit PNG format in C," 2019. Last visited on March,2019.

[205] S. Karl, "Smoothing Kinect Depth Frames in Real-Time," 2019. Last visited on March,2019.

[206] C. Kim, S. Yun, S.-W. Jung, and C. S. Won, "Color and Depth Image Correspondence for Kinect v2," in *Advanced Multimedia and Ubiquitous Engineering* (J. J. J. H. Park, H.-C. Chao, H. Arabnia, and N. Y. Yen, eds.), (Berlin, Heidelberg), pp. 111–116, Springer Berlin Heidelberg, 2015.

[207] Stereolabs, "ZED Stereo Camera," 2017. Last visited on 2019.

[208] S. Zhou, J. Zhang, W. Zuo, H. Xie, J. Pan, and J. S. Ren, "DAVANet: Stereo Deblurring with View Aggregation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10996–11005, 2019.

[209] C. Zhao, B. Fan, J. Hu, Z. Zhang, Q. Pan, and X. Wang, "Disparity Map Enhancement based Stereo Matching Method Using Optical Flow," in *2018 IEEE 14th International Conference on Control and Automation (ICCA)*, pp. 69–74, 2018.

[210] T. Gupta and H. Li, "Indoor Mapping for Smart CitiesAn Affordable Approach: Using Kinect Sensor and ZED stereo camera," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–8, 2017.

[211] M. T. Tran, D. H. Kim, C. K. Kim, H. K. Kim, and S. B. Kim, "Determination of Injury Rate on Fish Surface Based on Fuzzy C-means Clustering Algorithm and L* a* b* Color Space Using ZED Stereo Camera," in *2018 15th International Conference on Ubiquitous Robots (UR)*, pp. 466–471, 2018.

[212] V. Varma, S. Adarsh, K. Ramachandran, and B. B. Nair, "Real Time Detection of Speed Hump/Bump and Distance Estimation with Deep Learning using GPU and ZED Stereo Camera," *Procedia computer science*, vol. 143, pp. 988–997, 2018.

[213] Y. Rahul and B. B. Nair, "Camera-Based Object Detection, Identification and Distance Estimation," in *2018 2nd International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE)*, pp. 203–205, IEEE, 2018.

[214] Y.-C. Tsai, K.-H. Chen, Y. Chen, and J.-H. Cheng, "Accurate and Fast Obstacle Detection Method for Automotive Applications Based on Stereo Vision," in *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1–4, 2018.

[215] A. Inc, "iPhone X," 2019. Last visited on March,2019.

[216] S. Zhang, "High-speed 3D Shape Measurement with Structured Light Methods: A review," *Optics and Lasers in Engineering*, vol. 106, pp. 119–131, 2018.

[217] A. Chandrasekhar, K. Natarajan, M. Yavarimanesh, and R. Mukkamala, "An iPhone Application for Blood Pressure Monitoring via the Oscillometric Finger Pressing Method," *Scientific reports*, vol. 8, no. 1, p. 13136, 2018.

[218] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based Feature Tracking with Probabilistic Data Association," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4465–4470, 2017.

[219] X. Zhao, K. Ri, and N. Wang, "Experimental Verification for Cable Force Estimation Using Handheld Shooting of Smartphones," *Journal of Sensors*, vol. 2017, pp. 1–13, 2017.

[220] W. Yin, Y. Ming, and L. Tian, "A Face Anti-spoofing Method Based on Optical Flow Field," in *2016 IEEE 13th International Conference on Signal Processing (ICSP)*, pp. 1333–1337, 2016.

[221] C. Kim, P. Chiu, and S. Chandra, "Dewarping Book Page Spreads Captured with a Mobile Phone Camera," in *International Workshop on Camera-Based Document Analysis and Recognition*, pp. 101–112, Springer, 2013.

[222] T. Brox and J. Malik, "Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 3, pp. 500–513, 2010.

Curriculum Vitae

Name:	Seereen Noorwali
Post-Secondary Education and Degrees:	Umm AlQura University Computer Science Department Makkah, Saudi Arabia 2005-2009 B.Sc (Major Computer Sciences) CultureWorks ESL London ON, Canada 2010-2011 University of Western Ontario Computer Science Department London ON, Canada 2011 - 2013 Master of Science
Honours and Awards:	Excellent and 1st Degree Honour in Computer Science Awards Department Umm Al-Qura University, Makkah 2009
Related Work Experience:	Teaching Assistant Umm Al-Qura University, Makkah 2009 - 2010 Lecturer in Computer Science department Umm Al-Qura University, Makkah 2013-2014 Teaching Assistant Western University 2014-2020